REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1996	PhD Dissertation

4. TITLE AND SUBTITLE
A Crosstalk Correcting Router that Uses Online Noise Simulation to Route High-Speed Multichip Modules

5. FUNDING NUMBERS

6. AUTHOR(S)
Kenneth J. McClellan, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/DS/ENG/97-16

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Alan Tewksbury
WL/AADI
WPAFB OH 45433-7333

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved For Public Release; Distribution Unlimited.

12b. DISTRIBUTION CODE

ABSTRACT (Maximum 200 words)

Existing MultiChip Module (MCM) auto-routers either ignore crosstalk or use over-simplified crosstalk approximations. Of the routers that do consider crosstalk, very few have the capability to correct crosstalk problems after they occur.

This dissertation describes a router that not only has an internal crosstalk model for high speed MCMs which is more accurate than that of other existing routers, but also has the capability to use an online simulator to more accurately determine noise levels. After crosstalk problems occur, the router has the ability to correct these problems. Through memory usage reductions and routing efficiency improvements, this maze router has proven to be a good alternative to the current method of manual routing for very large, high-speed MCM designs.

DTIC QUALITY INSPECTED 8

14. SUBJECT TERMS
Routing, Multichip Modules, Crosstalk, Noise, VLSI

15. NUMBER OF PAGES
125

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

AFIT/DS/ENG/97-16

A CROSSTALK CORRECTING ROUTER THAT USES ONLINE NOISE SIMULATION TO ROUTE HIGH-SPEED MULTICHIP MODULES

DISSERTATION

Kenneth J. McClellan, Jr., Captain, USAF

AFIT/DS/ENG/97-16

AFIT/DS/ENG/97-16

A CROSSTALK CORRECTING ROUTER THAT USES ONLINE NOISE SIMULATION TO ROUTE HIGH-SPEED MULTICHIP MODULES

DISSERTATION

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Kenneth J. McClellan, Jr., B.S.E.E., M.S.E.E.
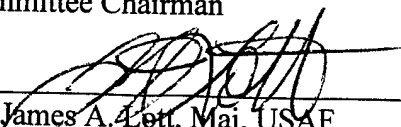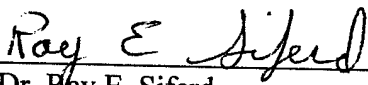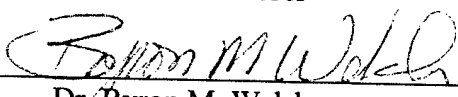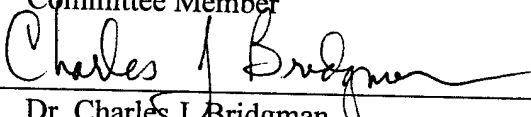
Captain, USAF

March 1997
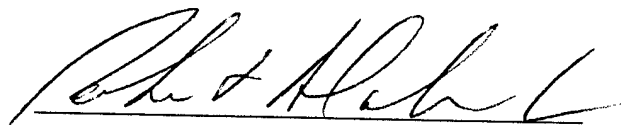
A CROSSTALK CORRECTING ROUTER THAT
USES ONLINE NOISE SIMULATION TO ROUTE
HIGH-SPEED MULTICHIP MODULES

Kenneth J. McClellan, Jr., B.S.E.E., M.S.E.E.
Captain, USAF

Approved:

_____   27 Feb 97
Dr. Tom S. Wailes, Lt Col, USAF
Committee Chairman

_____   27 Feb 97
Dr. James A. Lott, Maj, USAF
Committee Member

_____   27 Feb 97
Dr. Ray E. Siferd
Committee Member

_____   27 Feb 97
Dr. Byron M. Welsh
Committee Member

_____   27 Feb 97
Dr. Charles L. Bridgman
Dean's Representative

Accepted:

_____
Dean, School of Engineering

ii

__Acknowledgments__

There are many people I would like to thank for their support throughout the course of this dissertation. I certainly cannot list them all here, so I will just list those to which I owe particular thanks.

First, I would like to thank my parents. Their love and support throughout my entire life is more than I could ever repay. I entered the doctorate program more for them than I did for myself. I have always tried to make them proud, and I would like to dedicate this dissertation to them.

I also would like to thank my wife Heather. I have been somewhat scarce around the house in this, our first year of marriage. I know that I have a lot of making up to do.

Of course I need to thank my advisor Lt Col Wailes. He kept me on the right track, found potential problem areas before they occurred, and helped to keep me motivated to finish.

Dr. Paul Franzon at North Carolina State University probably gave me the most help in defining and understanding my area of research. To Dr. Franzon, I am greatly indebted.

Finally, I would like to thank the rest of my research committee: Dr. Byron Welsh, Dr. Ray Siferd, and Maj James Lott. They all took time from their busy schedules to support my research.

Table of Contents

List of Figures

List of Tables

AFIT/DS/ENG/97-16

__Abstract__

Existing MultiChip Module (MCM) auto-routers either ignore crosstalk or use over-simplified crosstalk approximations. Of the routers that do consider crosstalk, very few have the capability to correct crosstalk problems after they occur.

This dissertation describes a router that not only has an internal crosstalk model for high speed MCMs which is more accurate than that of other existing routers, but also has the capability to use an online simulator to more accurately determine noise levels. After crosstalk problems occur, the router has the ability to correct these problems. Through memory usage reductions and routing efficiency improvements, this maze router has proven to be a good alternative to the current method of manual routing for very large, high-speed MCM designs.

A CROSSTALK CORRECTING ROUTER THAT USES ONLINE NOISE SIMULATION TO ROUTE HIGH-SPEED MULTICHIP MODULES

I. Introduction

Currently, the routing of high speed MultiChip Module (MCM) designs is done using one of three different methods. One way is to use an auto-router to route the design, simulate the design to find potential crosstalk problems, and then manually fix the problems. Using this methodology, the MCM design needs to be re-simulated to check for problems after each manual change to the routing. Thus, this process is iterative and very time consuming.

Another method to route high speed MCMs is to use an auto-router that uses crosstalk constraints to drive the routing. The problem with this method is that most of the crosstalk-driven auto-routers use conservative preset spacing rules to avoid crosstalk problems. This potentially wastes precious space on the MCM substrate. In addition, all of the current auto-routers use simplified pair-wise crosstalk models that are not very accurate at high frequencies. This often results in manual rerouting of the design late in the design cycle.

The final method being used to route MCMs is to manually route the entire design. Some companies (e.g. the Mayo Foundation) have decided that none of the available

auto-routers perform adequately for high speed MCMs and have chosen to manually route every design. This is obviously very time consuming. However, it may not be as time consuming as trying to fix the problems that a poor auto-router creates.

Due to ever increasing clock speeds, crosstalk-driven auto-routers have become an increasingly popular research topic. This dissertation contributes to this body of research by applying new, more complex algorithms to crosstalk-driven routing.

1.1 Definitions

Definition 1. *A* **channel** *is a partition of routing area.*

Generally, channels are defined by a channel grapher that partitions the routing area into smaller, more manageable pieces. These pieces are called channels .

Definition 2. *An* **edge** *is a boundary between two channels.*

Channels are rectangular, thus each channel has four sides. If all channels are not the same size, a channel can have more than four edges, however. In this case, each side of the rectangle is divided into multiple edges. For example, the channels are not all the same size in Figure 1. Channel 4 has four sides (like any rectangle), but it has six edges. Channel 4 has two edges each on both the top side and bottom side of the rectangle.

Definition 3. *An* **overflow** *is a net that an auto-router failed to route.*

When an auto-router has finished, there are usually a few nets that were not routed for various reasons. These nets are called overflows. Overflows are generally caused by

congestion in a channel. Overflows are very common to auto-routers and must be routed manually.
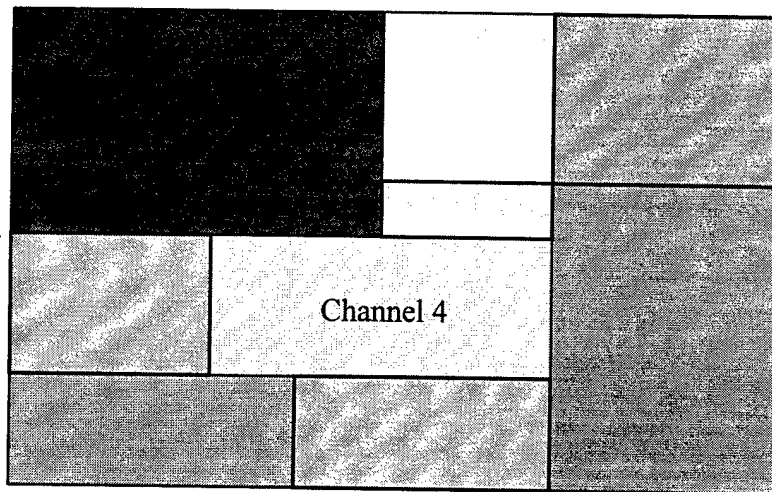
Figure 1. Possible Channel Partitioning.

Definition 4. Coupling *is the electrical interaction between two or more conductors.*

Coupling occurs anytime two conductors are in proximity and have a non-zero projection on one another. The electromagnetic fields of one conductor induces a current in the other conductor. The amount of coupling between two conductors is measured by the mutual capacitance and mutual inductance of the lines.

Definition 5. Crosstalk *is the induced signal in one line due to the activity of another.*

A change in voltage of one line (such as a clock edge) produces a change in the electromagnetic field of a conductor. These field changes produce currents and therefore

voltages in any conductors with which the line is coupled. The conductor that has the induced signal then in turn induces another signal on the original conductor. The size and shape of the induced signal is a function of the distance between the conductors, the coupled length, the material properties, and the rate of change of the voltage.

Definition 6. *A **discontinuity** is any change in impedance in a signal path.*

The most common discontinuities are vias and bends in the conductor. Discontinuities are important because they cause signals to be reflected.

Definition 7. *A **net** is a collection of conductors that are electrically connected.*

Terminals and the wiring between those terminals are all part of a net. A net is not limited to two terminals. Bus lines are good examples of nets that have many terminals.

Definition 8. Far-end *refers to the receiver end of a net.*

Crosstalk values are commonly computed at the ends of a net. Thus, the far-end crosstalk is the resulting noise induced in a conductor seen at the receiver end of the net.

Definition 9. Near-end *refers to the driver end of a net.*

Like the term far-end, near-end is commonly used when speaking about crosstalk. The near-end crosstalk is the resulting noise induced in a conductor seen at the driver end of the net.

Definition 10. *An* **interconnect** *is the electrical connection between a driver on one die and the receiver on another die.*

Although the interconnects between dice are normally considered to include the bonding wires and pads, this research is only interested in the connections between the bonding pads.

Definition 11. *A* **multichip module (MCM)** *is two or more IC dice connected on a single, but usually layered, substrate to form a single package.*

MCMs grew out of the hybrid microcircuit market. There are many different types of MCMs including MCM-D, MCM-C, and MCM-L. The identifier after MCM (i.e. MCM-X) refers to the type of substrate used for the interconnections, whether it be dielectric material, ceramic, or laminate.

Definition 12. **Pair-wise** *refers to evaluating a system of interactions by evaluating two components at a time and adding the results.*

For example, calculate the crosstalk on line 1 induced by line 2. Then calculate the crosstalk induced on line 1 by line 3. Finally, add the two to determine the total crosstalk on line 1. This is a good approximation, but it is not the true value of the crosstalk on line 1.

Definition 13. *A* **reflection** *is the portion of a signal that reverses direction when it encounters a change in impedance.*

Reflections are caused by discontinuities in the electrical path. The larger the difference in impedance, the larger the portion of the signal that is reflected.

Definition 14. *The* **saturation crosstalk** *is the maximum amount of crosstalk possible between two conductors given their spacing, independent of their coupled length.*

The amount of crosstalk between two conductors increases linearly (approximately) with coupled length up to a maximum value. This maximum value occurs at a length defined as the saturation length and is called the saturation crosstalk. Once the saturation length has been reached, the amount of crosstalk does not increase as the coupled length increases.

1.2 Problem

Existing MCM Electronic Design Automation (EDA) tools either ignore crosstalk or use over-simplified net-pair spacing rules. If problems are detected, the routing step must be iteratively redone and tested until no problems are found. This process may require the designer to partially or totally redo the routing step manually.

In future designs, the use of 100 ps and faster edges will greatly increase crosstalk noise levels and improved methods for crosstalk noise control will be needed. For example, consider a 1 inch long net on an MCM. With a 5 volt rise time of 1 ns, the saturated crosstalk occurs at a coupled length of 3 inches. Thus the maximum crosstalk

6

noise that could occur in a dual-stripline is 0.6 V (assuming separation/height ratio is 1). However, with a rise time of 100 ps, the saturated crosstalk occurs at a coupled length of 0.3 inches, and the maximum possible crosstalk on the net is 5.4 V. So the importance of crosstalk noise control for faster clock rates is extremely important.

1.3 Scope

The scope of this research effort was confined to the design tools, fabrication processes, and test circuits that were available. In particular, the focus of this research was on MCM-Ds with operating frequencies in the range of 600 MHz - 2 GHz. The software used to simulate the circuits and calculate the crosstalk values was ContecSPICE by CONTEC Microelectronics U.S.A. Inc. This software produces highly accurate crosstalk values because it does not use closed-form approximations used by most other tools. In addition, Dr. Franzon, a noted expert in the field, uses ContecSPICE in his research at North Carolina State University. Thus, by using ContecSPICE, we have been able to compare results of collaborative research.

Several assumptions were made during the course of this dissertation. One assumption is that the routing tool developed by this research will be limited to MCM-D technology. MCM-D, in the opinion of many experts, is the MCM technology of the future. It has a much finer pitch than other MCM technologies making the routing denser and crosstalk problems more pronounced.

Another assumption is that the router will be limited to two-layer rectilinear routing. Assuming only two-layer routing is not considered a serious limitation to this research

because the majority of MCM-D designs only have two layers of routing. Also, most routers use x-y pairs to route designs. Therefore a two-layer router can easily be converted into an unlimited layer router and that router could then be used on any MCM design (MCM-C, MCM-L, etc.).

The routing layers in an MCM-D are typically either stripline or microstrip. This research will concentrate on microstrip geometries. Microstrip has lower delay (thus higher performance), higher crosstalk, and larger reflections. Therefore, crosstalk is more of a problem in microstrip MCMs than in stripline. Also, the microstrip geometry has been better characterized in the literature because of the work done with printed circuit boards.

This research assumes that the nets have equal line widths. This is generally the case of most digital designs.

In the case of the crosstalk calculations internal to the router, the electromagnetic fields are assumed to be in quasi-TEM mode. This means that the longitudinal component of the fields are assumed to be zero. The Telegrapher's equations are based on TEM mode conditions. Pan and Hwang have shown that the Telegrapher's equations can give good results up to frequencies of 10 GHz [42] [86].

The MCMs under design will be assumed to have solid ground planes to simplify crosstalk calculations. Most MCMs have meshed ground planes, because crosstalk in coupled interconnects with meshed or slotted ground planes is less than that for coupled interconnects with solid ground planes [67]. Therefore, the solid ground plane assumption will provide the worst case scenario for nets under route.

8

The reflections due to bends are assumed to be negligible. Since this router is a rectilinear maze router, all bends in a routed net will be at a 90° angle. Rainal [89] shows for a typical 90° angle that a maximum of 4.7% of an incoming signal is reflected. For example, assume that 0.1 V of crosstalk is induced on a line. When the crosstalk pulse travels past a bend, up to 0.0047 V is reflected into the opposite direction. It should also be noted that reflections from multiple bends are not additive because of the timing involved. Therefore, bends are negligible; however, the number of bends in each net will be minimized.

Based on a conversations with Dr. Hayes [40] and Dr. Franzon [34], vias are also negligible. Dr. Hayes says that the only time vias are worth considering is when they are in conjunction with a bend (which is most of the time). In such a case, the via should still be ignored and the bend taken into account. However, as discussed above, bends are also negligible. Because this router also minimizes vias through the inertia code discussed later, this research will not compute their contribution.

1.4 Approach

The approach of this dissertation was to verify that external software could be used to simulate circuits and the results of that simulation could then be used to correct crosstalk problems. This was done by developing a router, developing interfaces with various software packages and the router, and using available test cases to verify the concept.

The router was developed in three parts: the channel grapher, the global router, and the detailed router. The channel grapher was developed from scratch. The global router was

based on the code developed in [78]. Finally, the detailed router was loosely based on code developed at North Carolina State University as a quarter project by Dr. Rodney Thomas. He based his code on Lee's algorithm for detailed routing. All three parts of the router were coded in standard ANSI C programming language.

In order to test the router, several test cases were needed. The Mayo Foundation was kind enough to provide two examples in Cadence Allegro format. This precipitated a need for an interface between Allegro and the router developed in this effort. The interface was created using the SKILL language that is built into Allegro. This interface allows routing information to be converted to a format readable by the router, and also allows Allegro to read the routed design after completion. This code can be found in Appendix A.

As discussed earlier in this chapter, ContecSPICE was used to simulate the designs to find the maximum crosstalk on each net. ContecRLGC was used to calculate RLGC information, crosstalk coefficients, and signal propagation speeds. Interfaces for these software packages were also needed and were created as part of this effort.

Finally, in order to test and verify the entire software suite, test cases were needed. Apart from the two designs provided by the Mayo Foundation, two MCC designs and twenty randomly generated designs were used. MCC1 and MCC2 are two designs that have been used as benchmarks in the literature. The twenty randomly generated designs were created to test the limits of this router as well as provide additional test data.

1.5 Materials and Equipment

Several software packages and a computer system were required to complete this dissertation. The VLSI laboratory at AFIT provided the necessary software running on a network of SPARC 10s and SPARC 20s running SunOS Release 4.1.3_U1B. Cadence Allegro 10.0 and Contec 2.200.1 were available on this system. Finally, GNU 2.5.8 was the C compiler used on this system.

1.6 Sequence of Presentation

This chapter has given an introduction to this dissertation. Included in the introduction were definitions needed to understand the use of the terms and phrases contained in this dissertation. Also included in the introduction was a problem statement, a discussion of the scope of this research, the approach used in this research, and finally a list of materials and equipment used throughout the research.

Chapter 2 discusses general background information about MCMs. In particular, MCM classifications are discussed and compared.

Chapter 3 contains the results of the literature review. Other routers are explained, discussed, and compared.

Chapter 4 contains the methodology used in this research. This chapter contains a general system overview, a detailed discussion of the crosstalk model used, and a discussion of the router. Finally, this chapter discusses how the crosstalk model is used within the router.

Chapter 5 contains the results of the router. In particular, this chapter contains the results of the memory changes, the inertia code results, and a comparison with other routers in literature.

Chapter 6 contains the crosstalk results for the router. Included in these results are the results for the internal crosstalk calculations, the results of the reflection code, and the results of the correction code.

Chapter 7 discusses the contributions to this field of study made by this dissertation. These contributions are listed and discussed.

Chapter 8 lists the conclusions and recommendations of this dissertation. The conclusions are based on the results of the previous chapters. Several recommendations are made concerning future research in this area.

Appendix A contains the SKILL code listings. Two programs are included: the program that converts from Allegro to the router, and the program that converts from the router to Allegro.

Appendix B contains the entire code listing for the router. Since the entire code listing is extremely long, this appendix in most copies of this dissertation will only contain the information necessary to obtain the code.

Appendix C contains a user's manual for the router. The user's manual discusses in detail the correct procedures needed to operate the router. Command-line options, associated files, and interfaces with Allegro are all discussed.

II. Background

2.1 General MCM Information

Multichip modules are two or more Integrated Circuit (IC) dice connected on a single, but usually layered, substrate to form a single package (see Figure 2). MCMs grew out of the hybrid microcircuit market [58] and have several advantages over conventional IC subsystem designs. Many agree that the only way to achieve the clock rates of desktop systems significantly faster than 100 MHz is with MCMs [73].
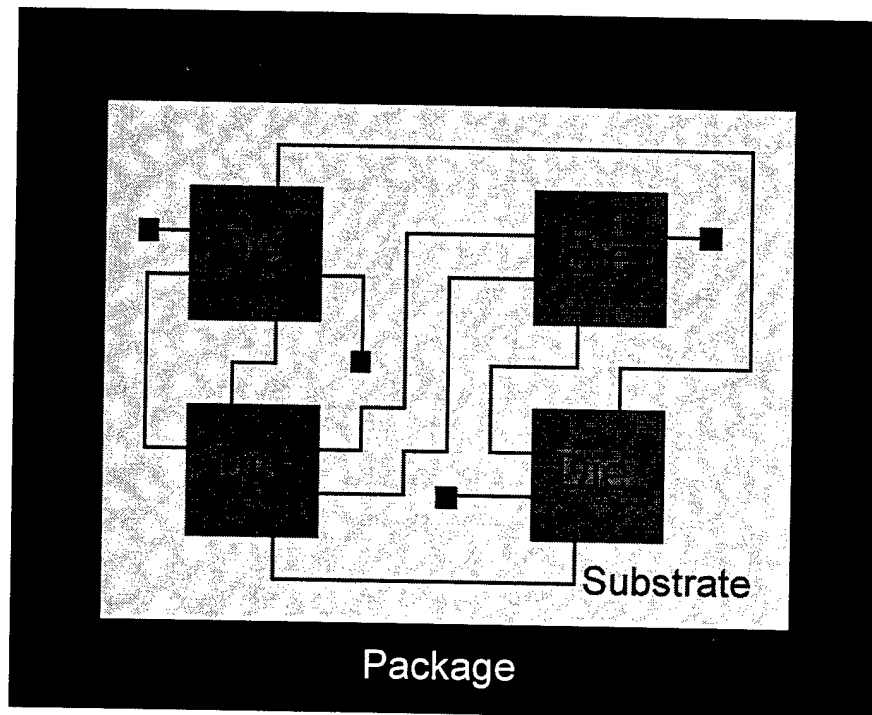
Figure 2. A Generic MCM.

Since the connections between dice in an MCM system is on a much smaller scale than on a printed-circuit board (PCB), the parasitics are much smaller (capacitance and

inductance) and the distance between the ICs is much smaller [58][112]. The smaller parasitics mean a faster rise and fall time of the signals resulting in faster apparent switching of on-chip buffers [112]. The smaller parasitics also generate less supply noise and ground bounce [112].

There are other advantages of MCMs other than the lower parasitics. The time it takes for signals to transit between ICs is less on MCMs due to not only the shorter distance, but also to the lower dielectric constants of the materials. Planar supply planes give MCMs better power distribution. In addition, MCMs generally have better thermal characteristics than IC subsystems. MCMs are also more rugged than other packaging techniques [51]. Finally, the worst-case physical parameters associated with fabrication are less likely to occur in MCMs than in PCBs. This means that the actual performance of an MCM will lean toward the fast end of the data sheet limits [112].

Along with these advantages come disadvantages. The cost of an MCM is generally higher than that of the separate ICs, but overall system cost is about the same in many applications. Also, testing and probing MCM subsystems is a challenge [52].

A major obstacle with MCMs is finding known good dies (KGD). Generally, vendors will not ship wafers with both good and bad die because it reveals too much about their processes. Instead, vendors ship bare untested dies. This creates difficult problems for MCM manufacturers. However, some companies have recently starting shipping tested KGDs [72].

2.2 Classifications of MCMs

There are many types and variations of MCMs. They can generally be classified by 1) die configuration, 2) metallization technology, and 3) substrate technology.

2.2.1 Die configuration

There are four main die configurations in MCMs. These are: 1) each die placed flat on the substrate, 2) layered dice, 3) dice on edge, and 4) dice stacked in layers [58].
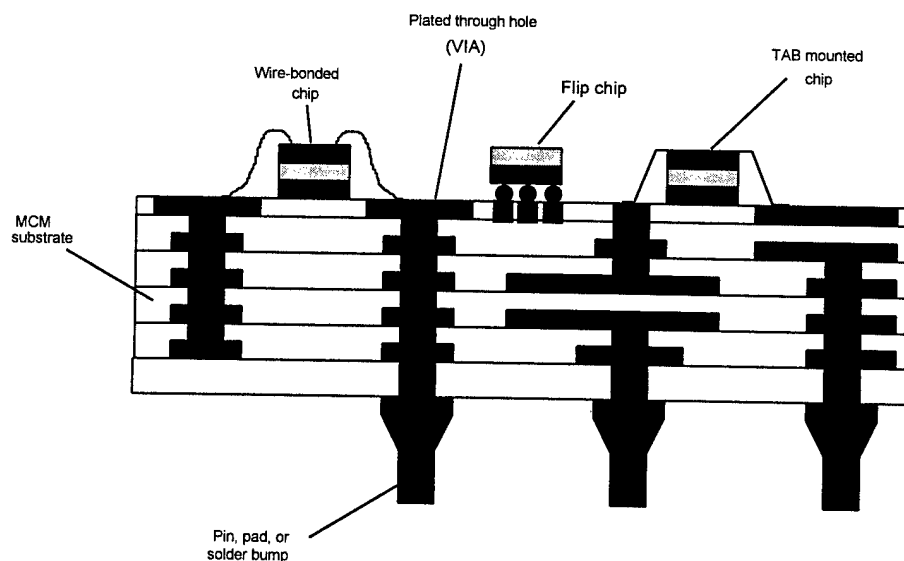
Figure 3. Typical Bonding Techniques.

Each die lying flat on the substrate is the most common configuration. This configuration is the least difficult and least costly configuration to fabricate. There are three main bonding techniques of this configuration (see Figure 3): wire bonding, flip-chip, and Tape Automated Bonding (TAB). Wire bonding is the chip interconnect method of choice because of the recent innovations of bonding equipment and packaging

15

technology [38]. However, Stan Drobac, vice-president of products at nChip, feels that the interconnect method of choice will eventually migrate to flip-chip due to higher packaging density and productivity. He expects this migration has already started, taking 5-10 years to complete [72]. Dimitry Grabbe, research director at AMP Inc., says that flip-chip is the most economical, highest-density permitting technology available [72]. Most feel, however, that TAB mounted dice will always have their place in the industry. Table 1 compares the bonding techniques in several areas.

Table 1. Comparison of Bonding Techniques [87].

Attribute	Wirebonding	TAB	Flip-chip
Minimum (and typical) I/O pitch (μm/mils)	100/4 (150/6)	50/2 (100/4)	250/10 (400/16)
Maximum I/O range (pins)	256-500	400-700	>700
Lead inductance (nH)	1-2	1	0.05-0.1
Mutual inductance between leads (pH)	100	5	1
Typical effective diameter (μm)	25	50	100
Connection technique	Perimeter. Array area is difficult, but possible using multi-height loops	Perimeter. Array area is moderately difficult, but possible using multi-layer tape	Perimeter and array area
Typical length (mm)	1	1	0.1
Packaging efficiency	Medium	Medium	High
Pretestability in fine pitch	Difficult with available instrumentation	Good	Difficult with available instrumentation
Ability to rework	Difficult	Fair	Good
Loop control	Fair	Good	Good
Dominant failure mechanisms	Fatigue, bond-pad corrosion	lead fatigue, interdiffusion	Fatigue, intermetallics
Flexibility of the manufacturing process	Excellent	Fair (gang bonding); Good (single-point bonding)	Good
Heat dissipation	Good (die bonded to substrate)	Good (die bonded to substrate)	Poor; Excellent if heat sink attached
Die availability	Excellent	Fair	Poor
Tool availability	Excellent	Fair	Fair
Technology maturity	Excellent	Good	Good
Market share	98%	<2%	<1%
Cost	Low	Medium	High (potentially low)

The layered-dice configuration is single unit with several layers fabricated on deposited silicon separated by thin dielectric layers. The layers are connected by vias

between the levels. Each layer is fabricated as a fresh wafer after the new crystalline layer of silicon has been deposited [58].

The dice stacked on edge configuration is a group of dice attached face-to-back to provide a flip-chip mountable block. This technique is mainly used for ultra-high package densities [58].

The dice stacked in layers configuration is similar to the dice stacked on edge configuration. The difference is that the dice are mounted to the substrate conventionally with tape automated bonds (TAB) and are stacked upon one another (instead of being on edge) [58].

2.2.2 Metallization Technology

MCMs either use a thick-film or a thin-film process in the fabrication of the metal layers on the substrate. Thick-film deposition is generally used only on ceramic substrates by screen printing and requires firing at high temperatures. Thin-film deposition is usually done by sputtering.

2.2.3 Substrate Technology

The most common way of classifying MCMs is by the type of substrate used. The most common substrates are laminate (MCM-L), ceramic (MCM-C), and dielectric (MCM-D).

MCM-L. MCM-L is based on laminated, multilayer printed circuit board technology, using copper for the wiring of the interconnects. Interlayer connections (vias) are done

by drilling. The number of routing layers for MCM-L is typically between four to ten, commonly six to eight [87].

MCM-C. MCM-C substrates use cofired ceramic or glass-ceramic technology with thick-film metallization. The typical configuration for MCM-Cs is with two adjacent signal layers that run orthogonal to each other (x-y pair), sandwiched between two meshed reference planes [44]. There are three basic ceramic-based technologies classified as MCM-Cs: thick-film multilayer (TFM), high-temperature cofired ceramic (HTCC), and low-temperature cofired ceramic (LTCC). TFM is the oldest of the three technologies and is processed layer by layer. Special pastes are deposited and patterned onto a ceramic substrate by screen printing. Unfired sheets of dielectric tape are used to make HTCC. The dielectric consists of glass fillers in a ceramic matrix. The ratio of ceramic to glass can be as high as 9:1. Commonly, tungsten or molymanganese is used as the conducting material. LTCC is similar to HTCC except that lower firing temperatures are achieved by using a ratio of ceramic to glass of 1:3. As a result of the lower firing temperatures, the conducting metal is usually copper, gold, or silver [87].

MCM-D. MCM-D is typically a thin-film process where the interconnections are formed by depositing dielectrics and conductors on a base substrate. The manufacturing processes are similar to those used in the semiconductor industry. Aluminum or copper is normally used as the conducting material [87]. Typically, MCM-D has two conducting layers. The two most common configurations for MCM-Ds are the stripline configuration and the microstrip configuration (see Figure 4). Table 2 compares the performance of these two transmission line types.

18
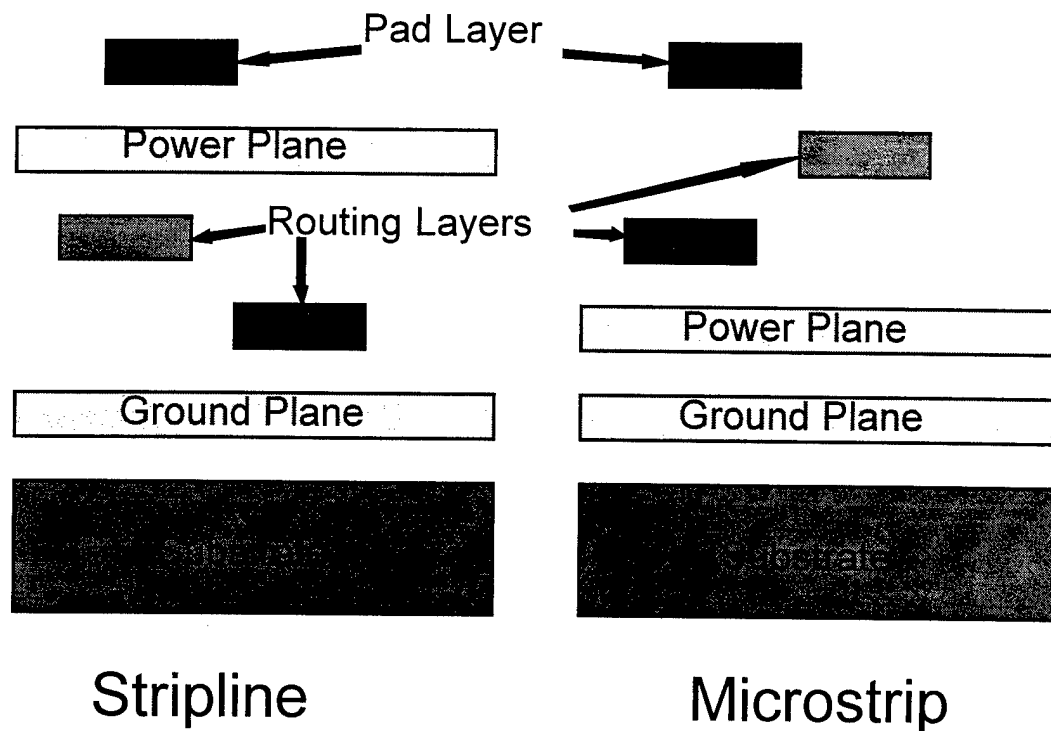
Figure 4. The Stripline and Microstrip Configurations.

The types of substrates are not limited to these three, however. The term MCM-Si

(Silicon) has been used to describe MCMs with a silicon substrate [58]. Also, researchers

are turning to synthetic diamond as a substrate because of its high thermal conductivity

(diamond has the highest thermal conductivity of any known solid material) [74].

The terms MCM-L, MCM-C, MCM-D, and the other MCM classifications have

become somewhat vague in recent times. Processing techniques that have been generally

associated with one type of substrate have also been applied to the others. In particular,

MCM-L and MCM-D materials and processes have been greatly mixed [72]. It has also

been found that using a ceramic substrate with MCM-D processes allows for some reworkability [70]. This may become a big advantage for that type of MCM.

Table 2. Performance of Stripline versus Microstrip [26].

	Loss	Reflections	Delay	Crosstalk
Stripline	HI	MED	HI	MED
Microstrip	MED	HI	LO	HI

2.3 MCM Substrate Technology Comparison

This section compares and discusses some of the more important attributes of MCM-L, MCM-C, and MCM-D. In particular, density, cost, and performance are compared and discussed. Table 3 compares those substrate technologies based on a more complete set of attributes.

In general, MCM-D is the most dense and MCM-L is the least dense technology. Density is related to the amount of wiring that the substrate can hold per unit area. A major reason for the difference in densities is in the available size of the vias. MCM-Ls have vias no smaller than 8 mils (some claim that anything smaller than 24 mils will not produce a decent yield [112]). MCM-Cs can have vias that are 4 mils and still have a good yield [72]. MCM-Ds may have vias smaller than 3 mils [112].

Table 3. MCM Substrate Technology Comparison [87].

	MCM-L	MCM-C		MCM-D	
		HTCC	LTCC	Ceramic metal	Silicon
Dielectric constant range	3.0-5.0	9.6	4.8-8.0	2.4-4.0	3.5-4.0
Maximum signal layers	15		50	5-10	5
Line width range (μm)	75-150	200-300	100-150	100-150	10-75
Spacing range (μm)	40-75	100-150	50-75	50-75	5-30
Via number range (μm)	200		125	25-75	15
Metal layer range	1-50		1-75	1-9	1-8
Thermal conductivity range (W/m-k)	<15		20-300		50-150
Typical maximum area (cm^2)	500	225	225	100	<100
Substrate I/O style	Peripheral and array	Peripheral and array	Peripheral and array	Peripheral and array	Peripheral
Tooling and setup costs	Low	Low	High	Medium	High
Vendor base	Large	Medium	Medium	Small	Small
Potential applications	Broad	Medium	Limited	Limited	Limited
Technology maturity	Good	Good	Good	Limited	Limited
Relative cost (low volume)	Low	Medium	Medium/ High	High	High
Relative cost (high volume)	Low	Medium/ High	Medium	Medium	High

The costs vary greatly between the types of MCMs. In general, MCM-Ls are the cheapest and MCM-Ds are the most expensive. The largest cost difference between MCM-Ls and MCM-Ds lie in the type of tooling required [112]. MCM-L film for masking cost $10/layer while the hard glass tooling for MCM-Ds easily cost $1000/layer [112]. However, this is a one time cost, for MCM-Ds. So for high volume production,

the price per unit will be greatly reduced. MCM-Ls require the drilling of every via, so additional vias increase the cost. This is not the case in MCM-Ds which use a lithography process to form the vias. Also, since MCM-Ds are more dense than MCM-Cs and MCM-Ls, less layers may be required to connect the dice. Therefore, the cost of MCM-Ds may actually be less than that of MCM-Cs and MCM-Ls in some applications. The same can be said for MCM-Cs compared to MCM-Ls.

Different processing techniques have been developed that have affected the relative cost of the MCM types. Pacific Microelectronics has unveiled a transfer tape process for MCM-C devices that it claims makes it a cost effective alternative to MCM-Ls and surface-mount circuit board technologies [27]. MCM-Si can be comparable in price to MCM-L and MCM-C by including passive devices in the substrate [71].

In general, MCM-L is last in density, performance, and reliability [81]; but, it is the least expensive. MCM-L/O (laminate/overlay), based on a thin-film high-density interconnect (HDI) for high performance, has recently been developed. The density of the MCM-L/O is greatly increased over MCM-L, but it still has a cost comparable to MCM-L [81].

The MCM-D has the best performance of the types of MCMs. An analysis was done that resulted in the selection of MCM-D for operation above 100-MHz clock speeds (compared to MCM-C and MCM-Si) [38]. MCM-L is used for much lower clock rates. It is felt by many that over the long term, the superior electrical performance and line geometry of MCM-D should assure its dominance over the other types of MCMs [81].

III. Related Work

There are many different MCM routers available. Most routers try to connect nets without any attempt at optimization; however, several attempt to optimize routing based on constraints such as delay, cost, net-length, or crosstalk. Table 4 shows a list of these routers and some of their important characteristics. Included in Table 4 are the constraints driving the routers. If the router supports crosstalk constraints, then the table also shows whether or not the crosstalk constraints are calculated using the pair-wise simplification, whether the crosstalk constraints are from calculations or simulations, and whether or not the router can correct crosstalk problems after they occur.

3.1 SLICE

SLICE is particularly useful for MCMs that have many routing layers. It is relatively fast (compared to other 3D routers), and it attempts to optimize the route base on wire length. SLICE gets its name from the fact that it works on only a "thin slice" of a two-layer routing grid at a time, thus reducing memory requirements. SLICE emphasizes planar routing; therefore, the number of vias created is generally less than that of other routers. In addition, SLICE processes many nets simultaneously. So, its solution is less dependent on net ordering. Khoo suggests that different routers may provide better solutions for MCMs with a small number of routing layers [54]. Also, SLICE does not support any constraint-driven parameters.

Table 4. Comparison of Existing Routers.

Router or Author	Routing Constraints	Pair-wise only?	Online Simulation/ Calculation?	Crosstalk Correcting?
SLICE	None	N/A	N/A	N/A
V4	# of Vias	N/A	N/A	N/A
SURF	Cost	N/A	N/A	N/A
YOR	# of Vias	N/A	N/A	N/A
	Yield			
Echelon	None	N/A	N/A	N/A
iPROMIS	Delay	N/A	N/A	N/A
WCSG	Net-Length	N/A	N/A	N/A
	Net-Shape			
Miyoshi	Parallel-Path-Length	Yes	None	No
CD3D	Net-Length	Yes	None	No
	Path-Separation			
	Parallel-Path-Length			
	# of Vias			
Allegro	Net-Length	Yes	None	No
	Path-Separation			
	Parallel-Path-Length			
	# of Vias			
Boardstation	Net-Length	Yes	None	No
	Path-Separation			
	Parallel-Path-Length			
	# of Vias			
Sengupta	Crosstalk	Yes	Calculation	No
Devaraj	Net-Length	Yes	None	No
	# of Vias			
Kirkpatrick	Crosstalk	Yes	Calculation	No
Chen	Crosstalk	Yes	Calculation	Reroute only
COP	Crosstalk	Yes	None	Yes
IS	Crosstalk	Yes	Simulation	Reroute only

3.2 V4

V4 was created by the same team that created SLICE. Essentially, it is the same type

of router as SLICE; however, V4 outperforms SLICE. V4 guarantees that no more than

four vias will be used to route a two terminal net, thereby reducing system costs and reflections. It is also 6.4 times faster than SLICE and has been shown to route a design in polynomial time. Like SLICE, V4 does not support any constraint-driven parameters.

3.3 SURF

SURF is a global router developed at the University of California. SURF uses the rubber-band algorithm. The rubber-band algorithm is an algorithm that initially assumes a direct connection. If there are obstacles blocking the connection, the connection is stretched around the obstacle. This process continues until no obstacles are blocking the connection. SURF is a true all-angle router using one-and-a-half layers. This means that as much routing as possible is done on a single layer and the second layer is only used for short "jumpers." Currently, SURF is being updated to include some limited cost optimization [21]. However, SURF does not account for potential crosstalk problems.

3.4 YOR

YOR (Yield Optimizing Routing algorithm) is a detailed routing algorithm developed to maximize fabrication yield. The YOR algorithm identifies and attempts to eliminate critical areas by floating, burying, and bumping net segments as well as shifting vias. The algorithm minimizes the number of vias used, which also increases yield [57]. Unlike the previous routers discussed, YOR identifies potential problems after routing and then attempts to correct them. However, YOR does not take crosstalk into account.

3.5 Echelon

Echelon is a detailed router that uses a novel grid construction scheme. Gridded routers generally make each layer the same size. However, the design rules are often different from one layer to another. So, other gridded routers make the grid spacing consistent with the layer that has the most restrictive design rules. This potentially wastes routing space on the other layers. Echelon uses a non-uniform grid so that each layer may have different resolutions. In this manner, each layer may be routed at the maximum resolution that the design rules allow [37]. However, Echelon does not optimize for crosstalk.

3.6 iPROMIS

iPROMIS (Illinois Performance-driven ROuter for Multichip InterconnectS) is an interactive MCM physical design tool developed at the University of Illinois. The auto-router portion of this tool uses a second-order transfer function of a lumped component transmission line model of the nets to determine delay. iPROMIS also has the capability to create SPICE files to simulate the nets. The routing is performed in two steps: tree generation and layer assignment. The tree generation step develops the transfer functions of the nets. The layer assignment step is designed to route many layers (common to MCM-Cs) while attempting to minimize delay [100]. Unfortunately, iPROMIS does not take crosstalk into consideration when routing.

3.7 WCSG

WCSG (Wiring Constraint Space Generator) is actually just one part of an overall routing system developed at North Carolina State University. No name has been given for the system, therefore WCSG will be used here for the sake of convenience. WCSG is based on the pre-characterization of nets to advise a global router. Several hundred sample nets are characterized and stored. When a global router considers placing a net, the rule generator advises the router which net shape and net lengths are appropriate [30][61]. This approach is similar to that of iPROMIS except that WCSG has a library of data from which it interpolates. The library is very time consuming to create, but this saves a significant amount of time during the routing phase. WCSG provides some wiring constraints for a detailed router, but a detailed router has not been developed for WCSG.

3.8 Miyoshi

The router described by Miyoshi was developed at Hiroshima University. It is a rectilinear router that keeps track of a net's total coupled length. If a placement of a segment puts the net over its budget, the placement does not occur [80]. This technique for crosstalk avoidance is common among routers, but is inadequate for high-speed designs. As discussed earlier, space on the substrate (i.e. cost) and delay times (i.e. performance) are adversely affected by this technique.

3.9 CD3D

CD3D (Constraint-Driven 3-Dimensional router) is a global thick-film MCM router. It was developed at Western Michigan University. CD3D routes the design with constraints on net-length, path separation, parallel-path-length, and the number of vias for each net. The values of these constraints are determined prior to routing [116]. These constraints help reduce any potential crosstalk problems. However, since these calculations are done before routing, precious space on the substrate may be wasted due to overly conservative routing rules. Path-length may also be slightly higher than needed which increases the signal transit time and therefore reduces system performance.

3.10 Allegro and Boardstation

Allegro (Cadence) and Boardstation (Mentor Graphics) are two commercial MCM auto-routing tools. Their operation appears to be similar to CD3D in the way the constraints are created and handled. As a result, both tools suffer from the same problems as CD3D. However, very little has been published on the operation of these routers so little is known for certain.

3.11 Sengupta

The tool described by Sengupta uses timing information known about the circuits to calculate and optimize for crosstalk. This router was created for printed circuit boards, but the concepts are general enough to be applied to MCMs. The peak voltages induced in a net are calculated based on when adjacent nets are active. Also, timing information about the noise on the net is taken into account. Given a noise budget, the spacings for

the nets are calculated and passed to the router. This tool was designed to prove a concept and only currently works on transmission line pairs of equal lengths [96].

3.12 Devaraj

The router described by Devaraj (developed at the University of Cincinnati) does the function of both a global and detailed router, but optimization occurs only at the detailed level. This router minimizes net-length, and allows the user to specify the maximum number of vias that can be used. Crosstalk is controlled by designating high frequency wires and attempting to maximize the distance between them within each channel [24]. This method of crosstalk control has several problems however. Maximizing the distance between high frequency wires will potentially increase path-lengths and therefore increase delay on these wires. These wires are probably the most critical wires in the design and therefore system performance will be adversely affected by the addition of delay. This method also does not guarantee that crosstalk limits will not be violated when many high frequency wires are present.

3.13 Kirkpatrick

The router described by D. A. Kirkpatrick is a crosstalk-driven channel router developed at the University of California/Berkeley. Crosstalk is controlled by calculating noise between wires and not allowing any wire to go over its noise budget. This router uses several techniques for crosstalk avoidance as it routes [56]. However, this router has several significant limitations. It is limited to channel routing and is not very useful for true detailed routing. Channel routing connects two sets of wires in a narrow channel.

Generally, there is a one-to-one connection between every wire in the two sets. The two sets of wires are just in a different physical order on opposite sides of the channel. Also, the crosstalk calculation methods are extremely primitive. Finally, the router makes over-simplifying assumptions about the geometries and couplings within the channel. For example, the router assumes that crosstalk occurs only between horizontal wires (x-axis wires). The coupling between vertical wires (y-axis wires) is ignored. This is done because the vertical wires tend to be very short compared to the horizontal wires in channel routing.

3.14 CHEN

This router was developed at IBM by Howard Chen and C. K. Wong. This router appears to be an excellent three-dimensional crosstalk constraint-driven router. This router determines the spacings between nets and maximum coupled lengths based on the noise budget remaining. This router also considers the crosstalk from nets on other layers. Perhaps the most unique aspect of this router is that it calculates the crosstalk between vias, which can be quite large in packages with a large number of layers. This router was built for MCM-C's (thick film process with many layers) [17]. All crosstalk calculations are pair-wise only. This is a good approximation at lower frequencies, but it is quite inadequate at very high frequencies. This router offers rip-up and reroute capability for nets that have exceeded their noise budgets. However, no other crosstalk correction techniques are available in this router.

3.15 COP

Crosstalk OPtimizer (COP) was created by Kyoung-Son Jhang of South Korea. COP is a detailed router that corrects crosstalk problems after the initial routing has been completed. Wire segments are swapped or pushed away from each other in order to reduce coupling capacitance [45]. This is a very effective approach to crosstalk correction. One problem with COP is that the way it calculates crosstalk is over-simplified. Not only does it assume pair-wise coupling, but it also ignores reflections from terminal mismatches and frequency issues.

3.16 Interconnect Synthesis

Interconnect Synthesis (IS), created by Interconnectix, may be the best crosstalk-driven MCM router commercially available today. It uses real-time analysis to simulate nets and determine spacing rules as it routes [75]. However, Interconnectix, for the most part, has not published the techniques and methods that IS uses to prevent crosstalk problems. Crosstalk analysis appears to be pair-wise only. Also, the only crosstalk correction technique available, if any, appears to be rip-up and reroute. Finally, the algorithms used for generating the routing advice appears to be over-simplified for high-speed applications [34].

3.17 Summary

This chapter discussed many different routers along with their strong points as well as with their weak. Their are several routers that account for crosstalk is some manner; however, the simplified routing rules and simplified crosstalk calculations used are

31

inadequate for high speed MCMs. The need exists for an router that can route high speed

MCMs at maximum density while guaranteeing that no crosstalk violations exist. In

order to do this, the crosstalk model used internally to the router must be much more

accurate than existing models. Also, the router must also be able to use an online

simulator to get maximum accuracy for crosstalk calculations. Finally, this router must

be able to efficiently correct any crosstalk violations when they are detected. The router

developed in this dissertation meets all these criteria.

IV. Methodology

4.1 Overview

The router developed by this research effort has many features that need to be

discussed. This introductory description is limited to general system components. A
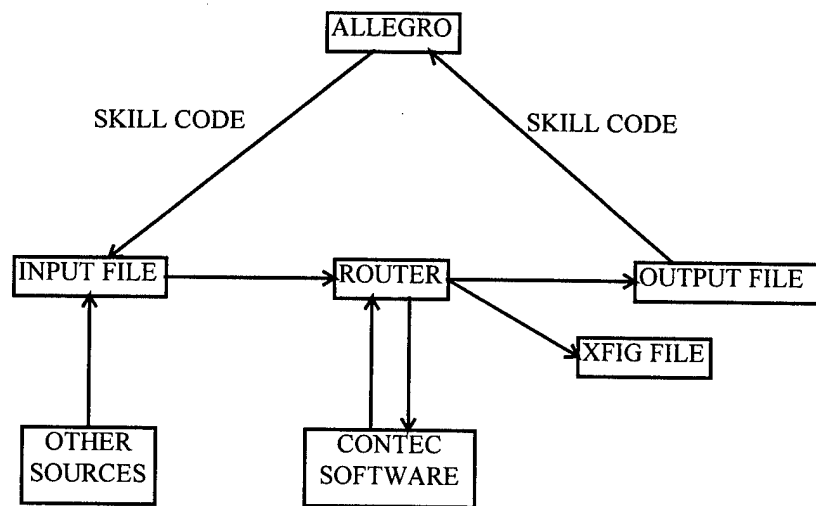
Figure 5. System Overview

more in depth discussion of the crosstalk model and the router are in following sections.

Figure 5 shows a simple diagram of the overall system architecture. The router

developed in this effort is in the center of the diagram. The input file for the router can

come from many different sources. Programs were written in SKILL code to interface

the router with Allegro. One program converts an Allegro design into the format required

by the router and writes it to a file. Another SKILL program reads the output from the router and adds it to a design in Allegro. The example MCC designs also needed to be converted. So, a C program was written to perform that conversion. The final way an input file may be created is by appending the output file to the original input file. This may be useful for special routing needs.

When crosstalk calculations are required in a given design, the router interfaces extensively with various Contec packages. ContecRLGC calculates the crosstalk coefficients (discussed later) and the resistance, inductance, conductance, and capacitance (RLGC) information. ContecLIF converts files generated by the router to a format readable by ContecSPICE. Finally, ContecSPICE simulates the circuit to find the crosstalk induced on each net.

The router creates two files, the output file and the Xfig file. The output file contains the information about the routed nets. This file may be read into Allegro or some other program. The Xfig file is a graphical representation of the routed design that is compatible with Xfig. Xfig is a publicly available graphics tool that is found on most UNIX systems. Xfig allows the user to view the routed design immediately after routing. The nets are color coded in Xfig to provide easier viewing. Also, the pins are marked as red dots.

4.2 Crosstalk Model

In order to maximize the density of a routed design, without violating crosstalk constraints, the crosstalk calculations must be extremely accurate. Ideally a simulator, in this case ContecSPICE, would be used for all the calculations needed. However, due to

34

speed considerations, real-time crosstalk calculations must be performed by the router. ContecSPICE is used as a final check on the noise levels. The following sub-sections discuss the crosstalk model used internally by the router in this research.

4.2.1 Coupling

Crosstalk occurs when the change in voltage in one conductor induces a voltage in a nearby conductor. This interaction is called coupling. Generally, all conductors are coupled if they are not shielded from each other. There are two types of coupling between lines: vertical coupling and horizontal coupling.
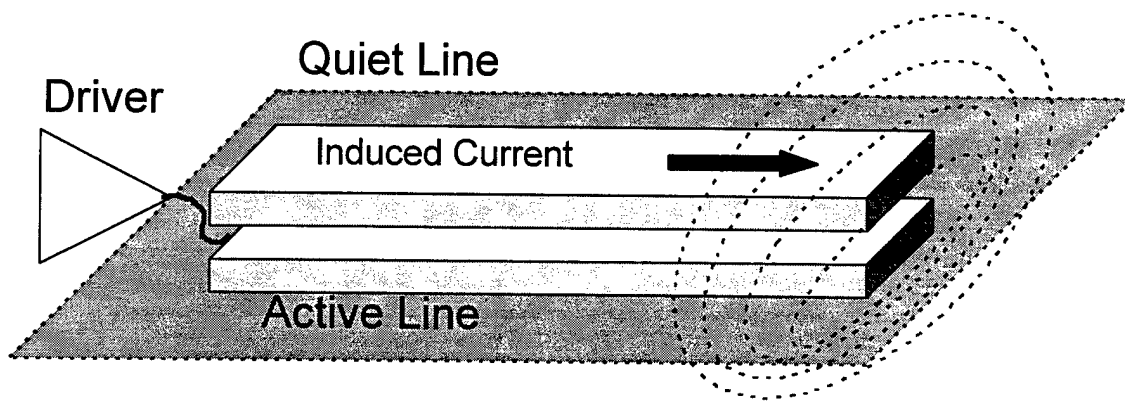
Figure 6. Vertical Coupling

Vertical coupling occurs when one conductor is directly over the other (see Figure 6). This situation is generally not allowed to happen when routing because this type of coupling generally produces the largest amount of noise. However, this dissertation will not forbid this situation from happening because it may increase density without diminishing performance, if done properly.

Horizontal coupling occurs when one conductor is side-by-side with the other conductor (see Figure 7). This is the more important of the two types of coupling because it occurs more often, and cannot be avoided.

It is possible to have both types of coupling between two conductors. Lines that are parallel but on different layers (and not directly above one another) have both types of coupling. Coupling of this sort is the least important because it is the weakest due to the distances involved.
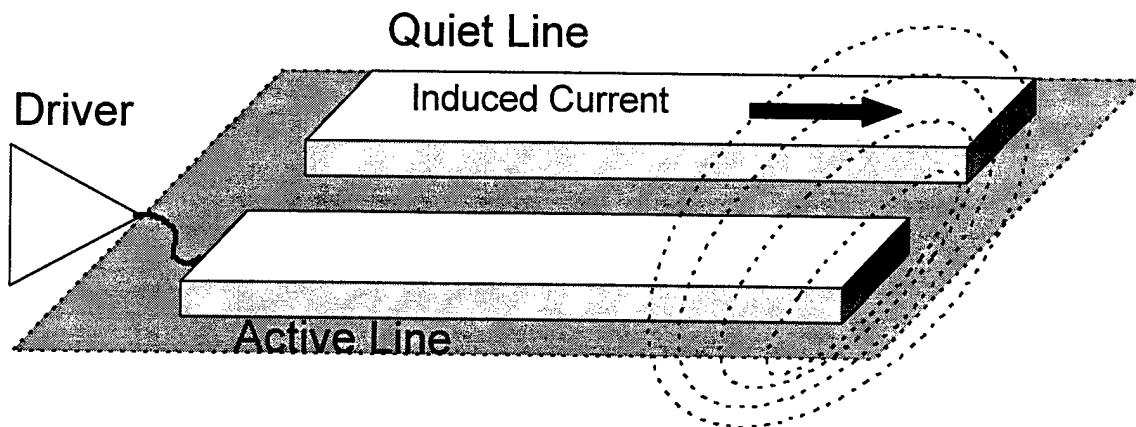
Figure 7. Horizontal Coupling.

Usually, only the nearest neighbor of any particular conductor needs to be considered when calculating crosstalk, because the effects of other conductors are negligible. However, for very high speed MCM systems this may not be true. Consider the bus in Figure 8. Line 3 is quiet while the other four lines are driven. Usually the coupling of lines 1 and 5 with line 3 is ignored. In fact, all of the existing crosstalk-driven auto-routers in literature ignore the effects from *next-nearest-neighbors* (lines 1 and 5).

In order to make the crosstalk calculations as accurate as possible, the router developed in this research includes not only the next-nearest-neighbors like lines 1 and 5, but also the next-next-nearest-neighbors like lines 0 and 6 (if they were shown). In many cases, the coupling of lines that far removed is not necessary, and the loss in router performance would be much greater than the value of the increased accuracy of the crosstalk calculations. For this reason, the coupling limits, within the router developed by this research effort, are dynamically determined at runtime based on the crosstalk coefficients, the input voltages, the size of the design, and the noise margins. The coupling limits for lines that are parallel, but on different layers, are also dynamically determined. Dynamically determining the coupling limits help balance the trade-offs between accuracy and speed.
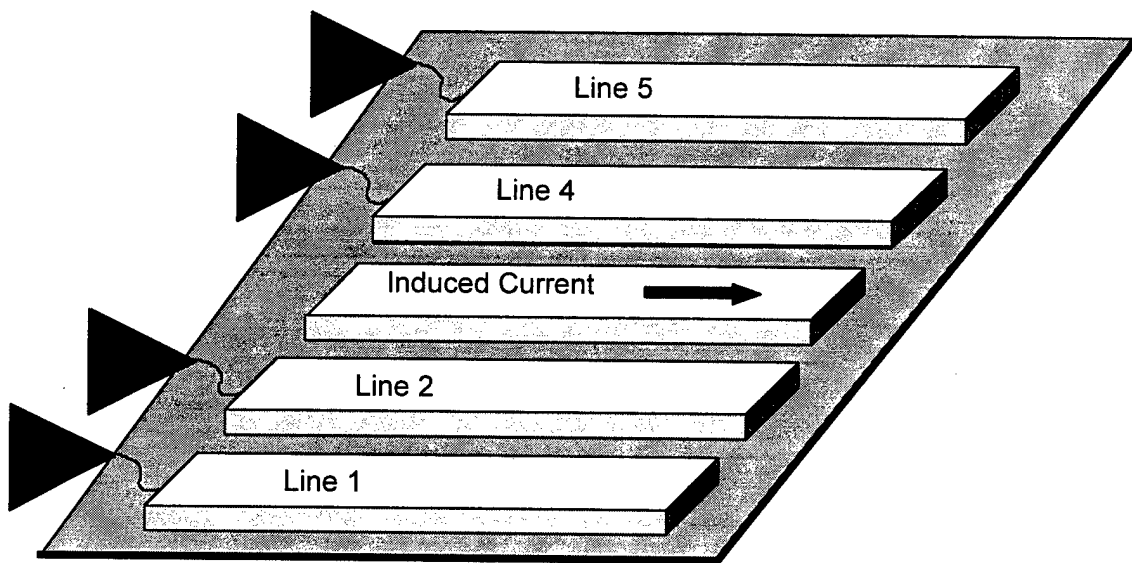
Figure 8. Five Conductor Bus.

Crosstalk is difficult to calculate because there exists no closed-form solutions. Nearly every CAD tool that calculates crosstalk, whether it is part of a router or not, uses

the quasi-TEM assumption. The quasi-TEM assumption is that the longitudinal components of the electric fields are negligible. From this assumption, the Telegrapher's equations can be derived. It has been shown that the Telegrapher's equations are valid up to at least 10 GHz [86][42]. The most general Telegrapher's equations are:

$$\frac{\partial}{\partial z}[V(z,t)] = -[R] \bullet [I(z,t)] - [L] \bullet \frac{\partial}{\partial t}[I(z,t)] \tag{1}$$

$$\frac{\partial}{\partial z}[I(z,t)] = -[G] \bullet [V(z,t)] - [C] \bullet \frac{\partial}{\partial t}[V(z,t)] \tag{2}$$

where $[I(z,t)]$ and $[V(z,t)]$ are current and voltage vectors, and $[R]$, $[L]$, $[G]$, and $[C]$ are the resistance, inductance, conductance, and capacitance matrices. For a three conductor system, $[R]$, $[L]$, $[G]$, and $[C]$ look like [68][90]:

$$[R] = \begin{bmatrix} R_{o1} & R_{m12} & R_{m13} \\ R_{m21} & R_{o2} & R_{m23} \\ R_{m31} & R_{m32} & R_{o3} \end{bmatrix}; [L] = \begin{bmatrix} L_{o1} & L_{m12} & L_{m13} \\ L_{m21} & L_{o2} & L_{m23} \\ L_{m31} & L_{m32} & L_{o3} \end{bmatrix}; [G] = \begin{bmatrix} G_{o1} & G_{m12} & G_{m13} \\ G_{m21} & G_{o2} & G_{m23} \\ G_{m31} & G_{m32} & G_{o3} \end{bmatrix}; [C] = \begin{bmatrix} C_{o1} & -C_{m12} & -C_{m13} \\ -C_{m21} & C_{o2} & -C_{m23} \\ -C_{m31} & -C_{m32} & C_{o3} \end{bmatrix} \tag{3}$$

where the X_{oj}'s (e.g. R_{o1} and G_{o1}) are the self values and the X_{mij}'s (e.g. R_{m21} and C_{m13}) are the mutual values.

Almost all CAD tools make approximations to solve the Telegrapher's equations from this point. Many assume that the system is lossless ($[R]=[G]=[0]$). However, for the frequencies of interest in this research, that is not advisable. Many assume that there is no dielectric loss (G_{mij}'s=0). This appears to be a valid assumption even at high frequencies. Hwang assumes no dielectric loss in his work up to a frequency of 10 GHz [42], which shows that the no dielectric loss assumption is valid at the frequency range of

38

interest in this research. Many CAD tools assume weak coupling, which means that the induced voltage and currents on the quiet line will not in turn induce voltage or current on the driving line. Appendix A in [55] discusses the mathematical significance of the weak coupling and no dielectric loss assumptions. Typically, if the no dielectric loss assumption is made, weak coupling is also assumed.

Katopis made the weak coupling and no dielectric loss assumptions when he developed his closed-form approximations to the crosstalk between low-loss transmission lines. These equations are discussed in [49] and [68].

Skin effect is definitely a concern at the frequency range with which this research was concerned. The Telegrapher's equations do account for the skin effect. The [R] and [L] matrices are functions of frequency when the skin effect is taken into account, but the [C] and [G] matrices are unaffected. Accounting for the skin effect, the crosstalk coefficients (K_{NE} and K_{FE}) become functions of frequency.

4.2.2 Crosstalk Calculations

The crosstalk calculations done by the router are based on the closed-form equations developed by Feller [29]. Feller's equations are compared to Katopis' equations in [68]. Feller's equations were used by the router because ContecRLGC provides the crosstalk coefficients used in the equations.

When a voltage is induced in a line, two voltage pulses (traveling in opposite directions) are created. The voltage pulse that travels towards the driver end of the driven line is called backward or near-end crosstalk. The pulse that travels away from the driver

end is called forward or far-end crosstalk. The two pulses are different in both shape and magnitude so separate equations are need.

The near-end crosstalk is given by:

$$V_{NE}(t) = K_{NE}\left[V_{in}(t) - V_{in}(t - 2t_d)\right] \tag{4}$$

where Vin(t) is the input voltage, td is the transit time for the signal to cross the coupled region, and KNE is the near-end coupling coefficient. KNE is also called the backward coupling coefficient and may be seen as Kb. KNE is given as:

$$K_{NE} = \frac{1}{4t_d}\left(\frac{L_m}{Z_o} + C_m Z_o\right) \tag{5}$$

where L_m is the mutual inductance, C_m is the mutual capacitance, and Z_o is the characteristic impedance of the lines.

The above equations are valid for both long and short lines. The peak value of the crosstalk depends on the length of the coupled region. The peak value is:

$$V_{peak} = \begin{cases} K_{NE}V_o, & \text{for } t_r \le 2t_d \\ 2t_d K_{NE}\dfrac{V_o}{t_r}, & \text{for } t_r \ge 2t_d \end{cases} \tag{6}$$

where t_r is the rise time of the input pulse and V_o is the peak input voltage.

The far-end crosstalk is given by:

$$V_{FE}(t) = K_{FE}\ell\frac{d}{dt}\left[V_{in}(t - t_d)\right] \tag{7}$$

where and K_{FE} is the near-end coupling coefficient. K_{FE} is also called the forward coupling coefficient and may be seen as K_f. K_{FE} is given as:

$$K_{FE} = -\frac{1}{2}\left(\frac{L_m}{Z_o} - C_m Z_o\right) \tag{8}$$

40

The peak crosstalk voltage is given by:

$$V_{peak} = K_{FE}\ell\frac{V_o}{t_r} \qquad (9)$$

where ℓ is the length of the coupled region.
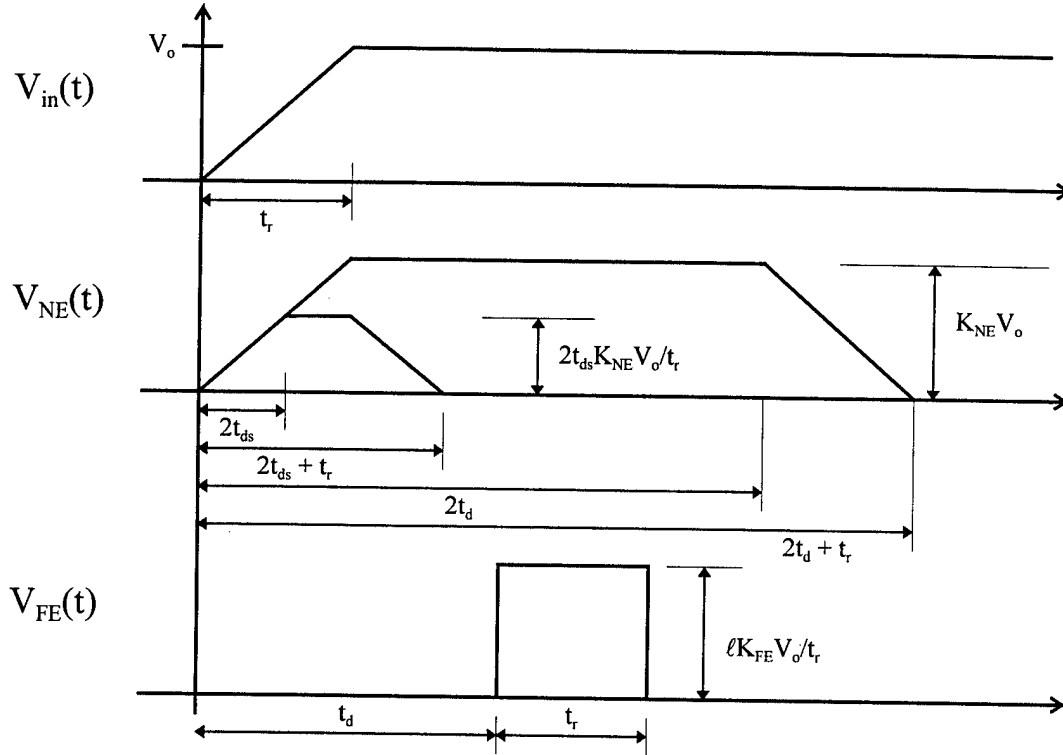
Figure 9. Forward and Backward Crosstalk Pulses [68].

Figure 9 shows the waveforms of both the backward and forward crosstalk pulses. $V_{NE}(t)$ has two pulses for the reasons discussed above. The smaller pulse is for short lines. The term t_{ds} represents t_d of the short line. The polarity of $V_{FE}(t)$ is assumed to be positive in the chart but is not necessarily the case.

As discussed in the previous sub-section, the effects of the coupling of lines that were further away than the nearest-neighbor were taken into account. One problem with trying

to calculate the crosstalk between two lines that have another line in between them is that the line in the middle partially shields them from each other.

In order to determine the amount of shielding, we start with the relationship $C_m \propto L_m$. From this relationship and the equations above, it can be shown that $V_{NE} \propto C_m$ and $V_{FE} \propto C_m$. Sakurai shows that the mutual capacitance per unit length can be approximated by the following equation [93]:

$$C_m = \varepsilon_0 \left[0.03 \frac{W}{H} + 0.83 \frac{T}{H} - 0.07 \left(\frac{T}{H} \right)^{0.222} \right] \left(\frac{S}{H} \right)^{-1.34}$$ (10)

where W, H, T, and S represent the distances shown in Figure 10.
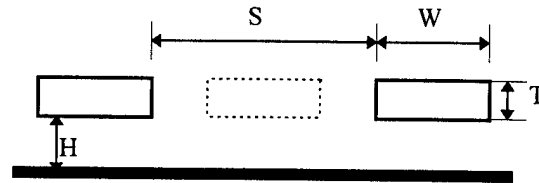
Figure 10. Wiring Geometry.

Assuming that the $0.07(T/H)^{0.222}$ term is negligible (a valid assumption), the following ratio can be derived:

$$\frac{C_{mwo}}{C_{mw}} = \frac{\varepsilon_{wo}}{\varepsilon_w} \left[1 + 27.27 \frac{T}{H} \right]$$ (11)

where C_{mw} is the mutual capacitance with the intermediate line present, C_{mwo} is the mutual capacitance without the intermediate line present, and ε_w and ε_{wo} are the dielectric constants with and without the intermediate line present, respectively.

42

Notice that the equation does not directly depend on distance. This means that the calculated crosstalk between any two wires that have another wire in between must be divided by the factor above. This is implemented in the router.

4.2.3 Reflections

Reflections are due to changes in impedance in a signal path and are a large source of noise in high speed systems. In most cases, the terminals of a net have a different impedance than the net itself. If the terminals are not impedance matched with the signal wire, then large reflections can occur and must be taken into account. The following discussion shows how this research accounts for and calculates reflections due to terminal impedance mismatches.

The lattice diagram method is used to account for reflections at the terminals. This method is used by North Carolina State University [10] [68] [96] and is considered to be very accurate. It is based on the superposition of the primitive pulses (in Figure 9) of the induced noise and the time of flight between terminals.
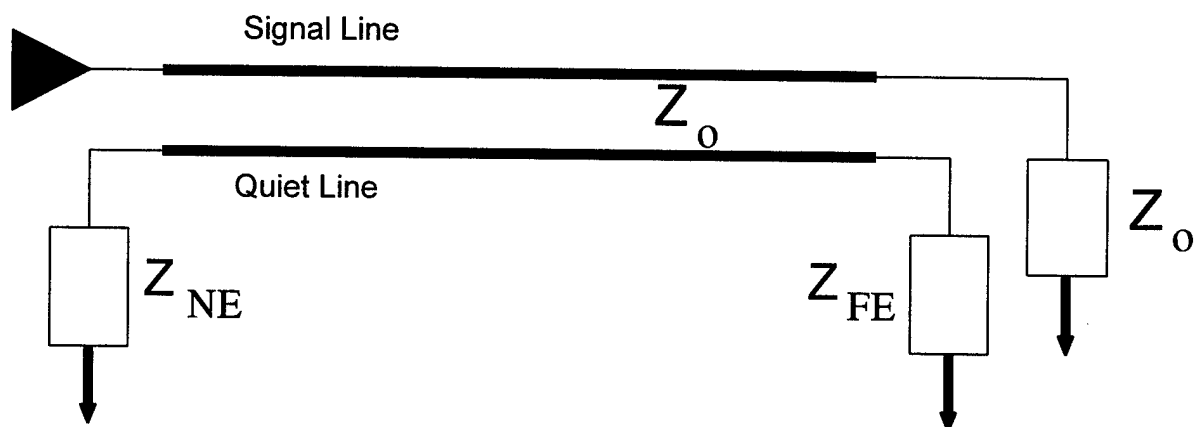
Figure 11. Two-conductor Transmission Line System.

Consider the two-conductor transmission line system in Figure 11. The quiet line has a characteristic impedance Z_o and near-end and far-end termination impedances of Z_{NE} and Z_{FE}, respectively. The reflection coefficients, Γ_{NE} and Γ_{FE}, can be calculated with the following equations:

$$\Gamma_{NE} = \left| \frac{Z_{NE} - Z_o}{Z_{NE} + Z_o} \right| \qquad (12)$$

$$\Gamma_{FE} = \left| \frac{Z_{FE} - Z_o}{Z_{FE} + Z_o} \right| \qquad (13)$$

The router allows the user to provide either the reflection coefficient, Γ, or the termination impedance and it will calculate Γ.
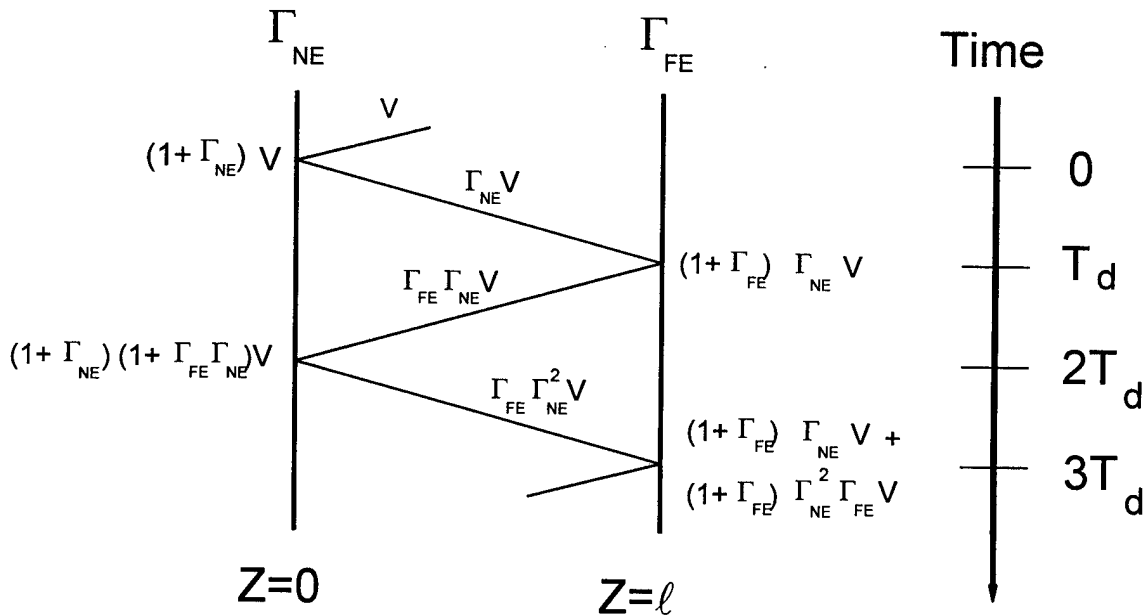
Figure 12. Lattice Diagram.

44

Figure 12 shows the lattice diagram for the near-end crosstalk pulse and its reflections. The far-end crosstalk pulse has a similar diagram. The reflections continue to contribute to the total noise at both ends of the net. The total noise at each end of the net is the sum of the contributions by both the near-end and far-end voltages and their reflections.

This makes the near-end total voltage to be [68] [96]:

$$
\begin{aligned}
V_{NE}^T(t) = & (1 + \Gamma_{NE}) V_{NE_o}(t) \\
& + (1 + \Gamma_{NE}) \Gamma_{NE} \Gamma_{FE} V_{NE_o}(t - 2T_d) + \dots \\
& + (1 + \Gamma_{NE})(\Gamma_{NE}^{i-1} \Gamma_{FE}^{i-1}) V_{NE_o}(t - 2(i-1)T_d) \\
& + \dots \\
& + (1 + \Gamma_{NE}) \Gamma_{FE} V_{FE_o}(t - T_d) \\
& + (1 + \Gamma_{NE}) \Gamma_{NE} \Gamma_{FE}^2 V_{FE_o}(t - 3T_d) + \dots \\
& + (1 + \Gamma_{NE})(\Gamma_{NE}^{i-1} \Gamma_{FE}^{i}) V_{FE_o}(t - (2i-1)T_d) \\
& + \dots
\end{aligned}
\tag{14}
$$

where $V_{NE_o}(t)$ and $V_{FE_o}(t)$ are the near-end and far-end primitive pulses which are calculated by the equations in the previous sub-section. T_d is the time of flight, the time it takes the signal to traverse the net, and is calculated by [96]:

$$
T_d = \frac{\ell \sqrt{\varepsilon_r}}{c}
\tag{15}
$$

where ℓ is the line length, and c is the speed of light. The far-end total voltage is very similar to Equation 14:

45

$$
\begin{aligned}
V_{NE}^{T}(t) = {} & (1 + \Gamma_{FE})V_{FE_o}(t) \\
& + (1 + \Gamma_{FE})\Gamma_{NE}\Gamma_{FE}V_{FE_o}(t - 2T_d) + \dots \\
& + (1 + \Gamma_{FE})(\Gamma_{NE}^{i-1}\Gamma_{FE}^{i-1})V_{FE_o}(t - 2(i - 1)T_d) \\
& + \dots \\
& + (1 + \Gamma_{FE})\Gamma_{NE}V_{NE_o}(t - T_d) \\
& + (1 + \Gamma_{FE})\Gamma_{FE}\Gamma_{NE}^{2}V_{NE_o}(t - 3T_d) + \dots \\
& + (1 + \Gamma_{FE})(\Gamma_{FE}^{i-1}\ \Gamma_{NE}^{i})V_{NE_o}(t - (2i - 1)T_d) \\
& + \dots
\end{aligned}
\tag{16}
$$

Obviously, these equations become quite cumbersome as the number of terminations increase. Therefore, an algorithm was developed as part of this research that calculates the crosstalk from reflections on a net with N terminations. That algorithm is shown in Figure **13**.

The algorithm works by calculating the initial crosstalk pulse shape, size, and time arrived at each termination. These pulses are then reflected to each of the other terminations with the new peak value and time arrived calculated. If the new peak value is below a cutoff value, that reflection is assumed to be negligible and is ignored. The cutoff value is calculated based on the noise limits and the initial value of the crosstalk pulse. The reflection process continues until all additional reflections are below the cutoff value.

The result of the above process is a list of pulses on each termination. Each pulse has a different start time and different peak value. Pulses also vary in shape as shown in Figure 9. The list of pulses is then combined by superposition and the maximum value is calculated.

```
for (i = 0; i < N; ++i)
        for (j = 0; j < N; ++j)
                time[i][j] = time[j][i] = distance(i,j)/signal speed
        events[i] = initial pulse    /* based on forward or backward pulse
                                        and distance from point of induction*/

done = FALSE;
while (!done)
        done = true;
        for (i = 0; i < N; ++i)
                event = events[i];
                event = find next unchecked event
                if (event != NULL)
                        noise = event->peak*GAMMA;
                        if (noise > cutoff)
                                for (j = 0; j < N; ++j)
                                        if (j <> i)
                                                add new event to events[j]

max_noise = 0.0
for (i = 0; i < N; ++i)
        noise = calc_peak_noise(events[i], Tr, Td);
        if (noise > max_noise)
                max_noise = noise;
```

Figure 13. Pseudocode for Reflection Algorithm.

4.2.4 Crosstalk Correction Techniques

Throughout the course of routing a design, crosstalk problems are likely to occur.

These problems can be avoided before the nets are placed or fixed after they are placed.

Most routers have chosen to attempt to avoid crosstalk problems. The router in this

47

research corrects the problems after they occur. Many recent papers have stated that this is the better of the two approaches [49][96][109].

Six crosstalk correction techniques were identified during the course of this research. These techniques are: push, level change, swap, shield, channel re-route, and rip-up. Figure 14 graphically shows the first three techniques.
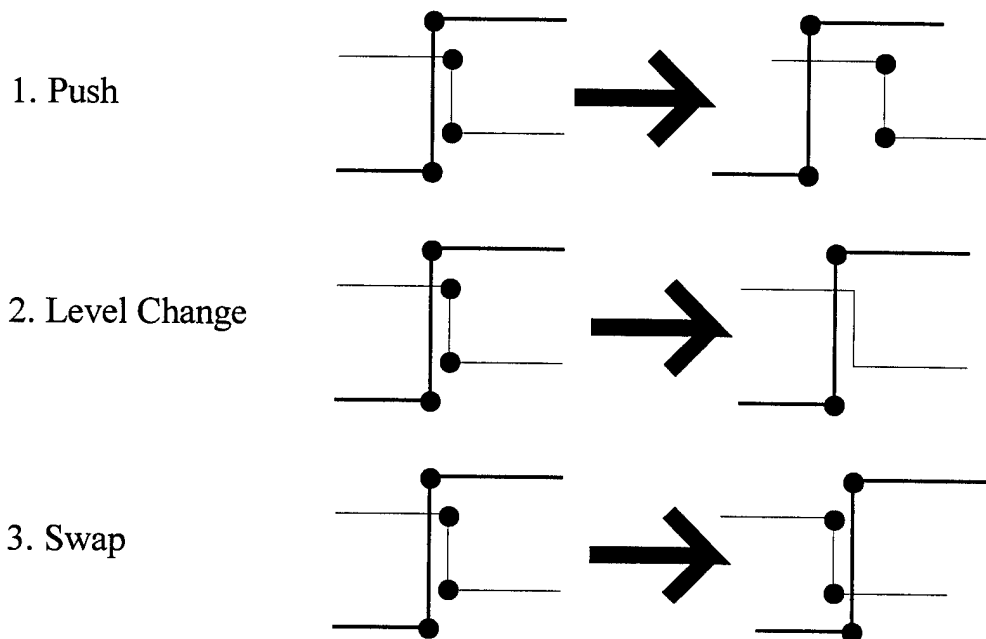
Figure 14. Crosstalk Correction Techniques.

The push technique is the most useful of all the techniques. The push technique simply moves the current net away from the problem net. The space between the nets is then blocked to prevent additional nets from creating the same problem in that area.

The level change technique is used only after all the nets in the channel have been routed. Normally, x-axis routing occurs on one layer and y-axis occurs on the other. The

level change technique moves a net segment from its original layer to the other. This not only reduces the crosstalk between the two nets, but it also allows the router to remove two vias. The only potential problem is that having an x-axis segment on the y-axis layer (or vice-versa) can greatly reduce the routing efficiency. That is why this technique is only allowed after the channel has been completely routed.

The swap is the least useful of the techniques. The adjacent segments of two different nets simply swap positions. This technique is least useful because the crosstalk between the two nets involved in the swap doesn't change. The swap is made under the condition that other nets in the area will induce less noise on the troubled net.

Another technique is to shield the nets by placing a small segment that is connected to a reference voltage between them. The effects of shielding in this manner is similar to that discussed earlier. The problem is that not all MCMs have reference planes. Since this technique is not valid for all MCMs, it was not implemented.

The channel re-route technique blocks the two segments with the most noise and attempts to re-route the net within the channel. If a new path is found that has less noise than the current path, the net is moved. This technique is attempted only after every net in the entire channel has been routed. This avoids duplicating the results of the other correction techniques.

The final crosstalk correction technique is rip-up. Rip-up is used as a last resort and can be disabled by a command line option. Rip-up permanently removes nets that violate the noise budgets. The removal of nets is done intelligently however. For example, assume nets A, B, and C all violate their noise budgets. If net B is removed, nets A and C

are now under their budgets. In this case, only one net needs to be removed to fix the crosstalk problem with three nets.

4.3 Routing

The routing step in MCM design is one of the most critical. A circuit may not be able to operate at designed speeds due to noise generated by crosstalk. This might render a design useless, depending on the application. This problem can be reduced by proper routing of the MCM design.

In general, routing is done in three steps: channel graphing, global routing, and detailed routing. In many cases, the channel grapher is part of the global router, but for this discussion they will be kept separate. The following sub-sections briefly describe these three parts of routing and the specific implementation used in this research.

3.3.1 Channel Grapher

The channel grapher partitions the entire design into areas called channels. Channels contain several routing layers, thus some papers refer to channels as towers because they are three dimensional [116]. Usually the channels are partitioned based on the pad layout and other obstacles to the routing such as thermal vias. In general the number of nets that can be routed through a given channel is also determined at this point.

This research used a global router written at North Carolina State University (NCSU). This router is named WCSG and was discussed in Chapter 3. A channel grapher was written to interface with WCSG.

A common problem with channel graphers is determining the number of nets a channel can hold [115] [92]. According to David Rowlands [92], one of the biggest problems encountered by channel graphers is obstacles in the channel (see Figure 15). As can be seen in Figure 15, only a few nets can be routed vertically in this particular channel. However, many more nets can be routed horizontally. Therefore the number of nets that can be routed through this channel depends on the direction the nets are being routed.
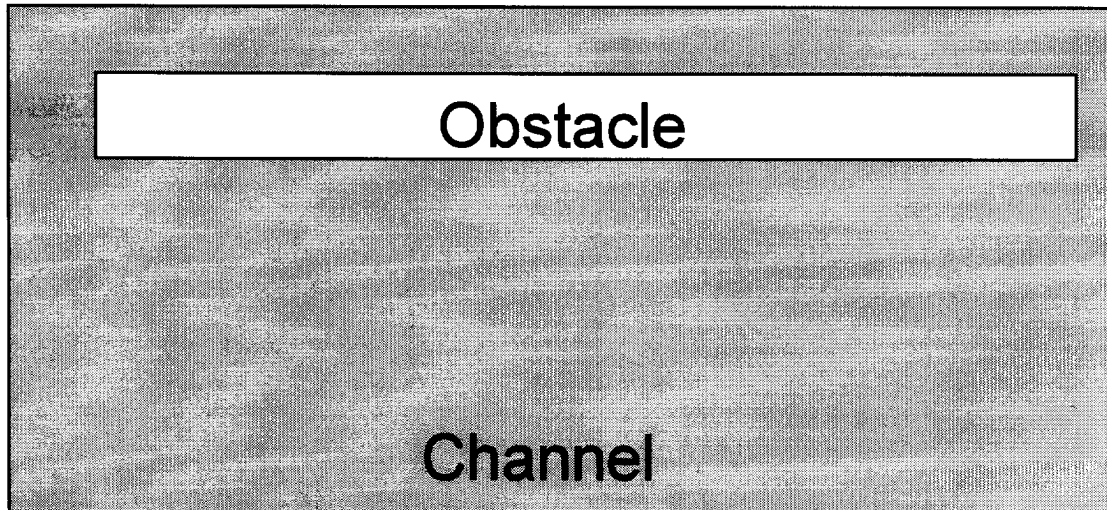
Figure 15. Channel with an Obstacle.

One solution to this problem is to have two variables defining the maximum number of nets through a channel: one for the horizontal-axis (x-axis) and one for the vertical-axis (y-axis). X-axis and y-axis routing are typically done on different layers, thus defining the channel capacities in terms of the x and y directions makes more sense than at first glance. The channel grapher in this research went one step further, capacities were

51

assigned to every edge of a channel. Therefore, four variables were created for each channel to keep track of the remaining capacity. As can be seen in the next section, the bookkeeping for this technique is quite simple.

4.3.2 Global Router

The second step is global routing. This step coarsely routes the entire design using the channels and edge capacities for the channels that were determined by the channel grapher. Congestion is detected and dealt with in this step.

As mentioned in the previous sub-section, the global router developed at NCSU, WCSG, is the foundation of the global router developed in this research. WCSG uses pre-characterization to help determine the net paths. Pre-characterization is a technique that requires a library of many different net shapes and sizes that have already been characterized through simulation or actual measurements. When a net of N nodes needs to be routed, the library is consulted to find an N-node net with a shape that is compatible with the location of the pads. All matches in the library are compared to find the best solution. Figure 16 shows four possible shapes of a three terminal net.

As discussed in the previous sub-section, this router uses separate limits on each edge for the number of nets allowed to be routed across that edge. The bookkeeping for separate edge capacities may actually be easier (or at least more accurate) than that of traditional methods. Since the global router is assigning a path for the different nets through the channels, the entry and exit edges of the channel for each net is known. So the cost of routing a net through a channel is just one for each edge it crosses.
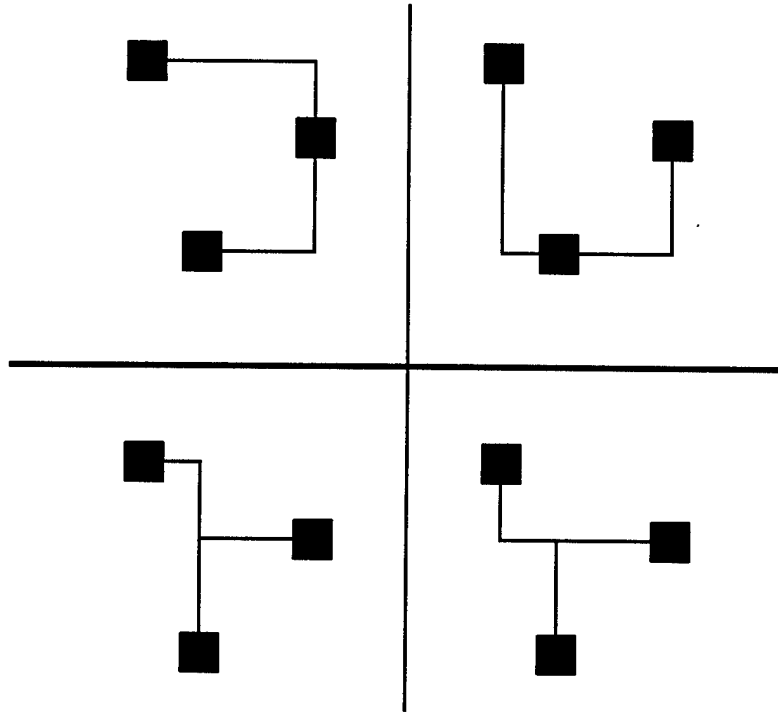
Figure 16. Net Shapes.

Many changes had to be made to WCSG. The data formats were changed to increase compatibility with the channel grapher and detailed router. Many data structures were also changed for memory reasons. WCSG used an extremely high amount of memory for designs with a large number of channels. Data structures were changed, deleted, or replaced by a function in order to decrease memory requirements. The results of these reductions are discussed in the next chapter.

One other change to WCSG was in the algorithm it used to determine the shortest path between channels. WCSG originally used Dikstra's algorithm. Due to the equal channel size assumption, Dikstra's algorithm was reduced and combined with Lee's algorithm. The resulting algorithm is similar to Lee's algorithm and has greatly reduced the processing time in this part of the code.

The global router finds several potential paths for each net. These paths, along with the edge capacities, are used to find the best combination of net paths for system performance. The paths are selected by a linear program solver.

4.3.3 Detailed Router

The exact location of each trace is not known until after the completion of the next step, the detailed routing. The detailed router does not route the entire design at the same time. It routes the design one channel at a time.

Once again, a program developed at NCSU was used as the foundation of the detailed router. Routing code that was developed as a quarter project by Dr. Rodney Thomas (a student at the time) provided a good start, but extensive re-writing was required to incorporate the additional features associated with this research.

As did the global router, the detailed router used extremely large amounts of memory. Many changes were done to reduce the memory requirements. The most significant of these changes was the creation of a *virtual grid*. If the grid for the detailed router is large, it can take up an excessive amount of memory. The size of the grid is based on the resolution required and the problem size. Therefore, large grid sizes are not uncommon. Instead of keeping the entire grid in memory at once, the virtual grid only has the active channel in memory. The other channels are recreated from the routing data as they are needed. In a large grid with 100k x 100k points, the original code required 320 Gbytes of memory and the new code requires only 13 Mbytes of memory. Additional results are in the next chapter. In order to allow the detailed router to look ahead into the next channel, the virtual channels slightly overlap.

Under certain circumstances, the original detailed router had a tendency to zigzag or stair-step from one point to the next. Not only did this create bends (and therefore increased reflected noise), but it also causes more vias. In order to fix this problem, new code was implemented which forces the route to continue in the same direction if it does not make the overall route longer. This new code is called *inertia code*. The inertia code reduces bends and vias which decreases the number of obstacles in the channel, thus increasing the routing efficiency.

4.4 Putting It Together

The key to this research effort was to incorporate the crosstalk calculations and correction techniques into the router such that the router routes the design based on the crosstalk constraints. The following describes how this was done.

The detailed router completely routes a net within the channel before starting on another net. If the detailed router cannot route a net within the channel, then the global router will attempt to re-route the net through different channels, and the detailed router will try again. If the global router cannot route a net, an overflow will occur and the net will need to be routed manually.

After the detailed router has placed a net in a given channel, the crosstalk to which it is subjected (within that channel) is calculated as discussed previously. Pointers to the nets imposing the crosstalk and the values of the crosstalk are stored. Also, the crosstalk that the newly routed net imposes on the other nets is also calculated and stored. Generally, two nets will impose the same amount of crosstalk on each other, but due to reflections the values may be different and are stored separately.

If the noise budget for the current net or a previously routed net is exceeded, then the detailed router attempts to correct the problem within the channel. Since pointers are kept to all sources of crosstalk, the location of the largest source of noise is known. The detailed router attempts to fix the problem at that point first using the techniques previously discussed. The spacing of the nets is the first solution attempted (the push technique). After attempting to push the segment, the router tries the swap technique and then the level change. If unable to get the net within its noise budget, the router tries the second largest source of crosstalk, and so on. If all of these attempts to reduce the crosstalk to within the noise budget fail, the net is re-routed; or, as a last resort, the net is permanently removed.

The detailed router watches for other insidious effects. One is that the crosstalk between two nets may already be saturated due to coupling in other channels. So the crosstalk between these two nets in the current channel will have no effect on the noise budget. The detailed router also considers obstacles and routed nets in the next channel before choosing where on an edge to end a net.

V. Router Results

5.1 Router Overview

In order to implement the crosstalk model and correction techniques developed in this research, the parts of the router that were discussed in the previous chapter had to be well integrated. The results in this chapter verify not only the respective portions of the code, but also the successful integration of all of the parts of the code. The following sections discuss the results from the memory usage changes, the results of the addition of the inertia code, and the comparison of this router with others in literature.

One of the most important tests for the router was a 9-chip design from the Mayo Foundation. This design was routed by a proprietary "high powered" router on the equivalent of a Cray-1. This took eight months, and the route still had errors. Mayo personnel were forced to route the design manually. This effort took eight weeks. Figure 17 shows the routed 9-chip design as it was routed by this router. The route took approximately 2.6 hours and 98.3% of the nets were completely routed in two layers. This information, along with the data throughout the remainder of this dissertation, proves this router to be a viable alternative to current methods.

5.2 Results of Memory Changes

There were several code changes that were aimed at reducing the memory requirements of the router. The following sub-sections discuss the results of these

changes. In particular, the results from the improvements to both the global and detailed

router are discussed.
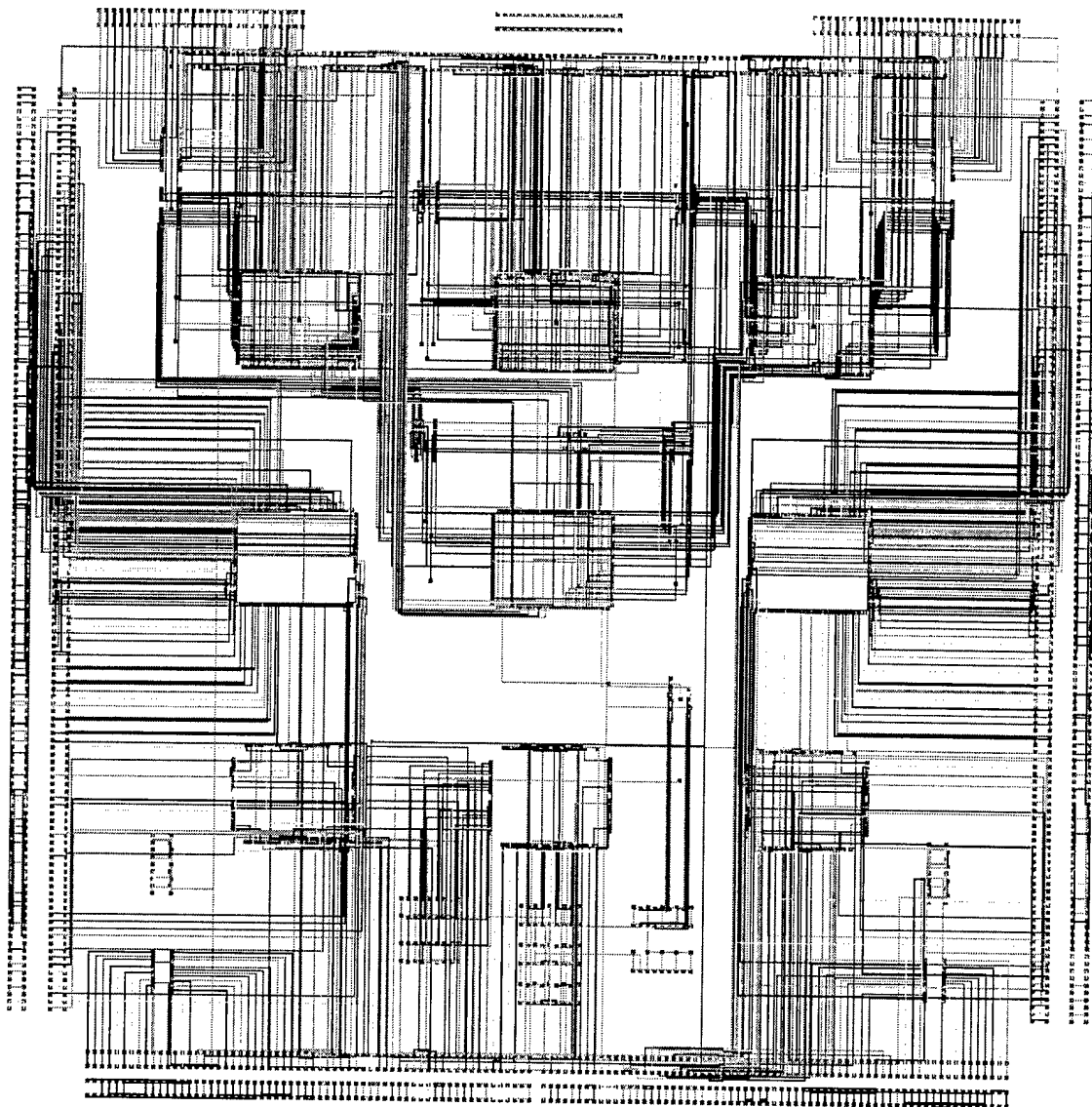
Figure 17. Xfig View of Routed 9-Chip Design.

58

5.2.1 Results of Global Router Memory Changes

Extensive rework of the WCSG code was required to remove memory intensive data structures and to remove memory leaks. These changes along with the changes allowed by the equal channel size assumption, provided a dramatic reduction in memory requirements by the global router. In fact, the memory requirements went from $O(n^2)$ to $O(n)$, where n is the number of channels.
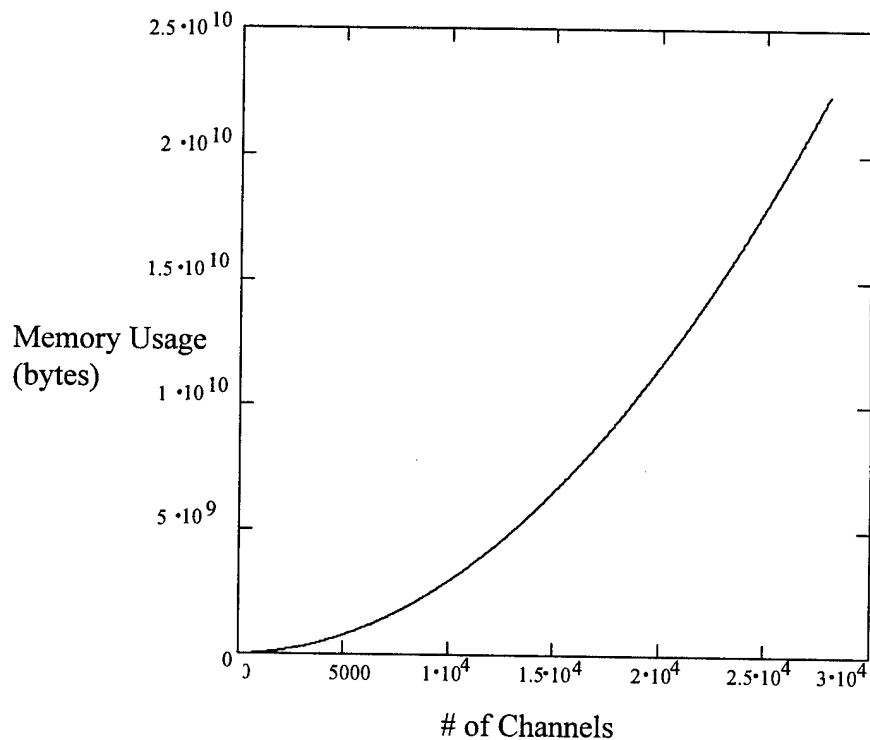
Figure 18. WCSG Memory Usage.

Figure 18 shows the memory usage of WCSG versus the number of channels. While 30 thousand channels (on the high side of the chart) may be somewhat unrealistic, the purpose of the chart is mainly to show the difference between the before and after

59

memory usages. Also, even though 30 thousand channels is somewhat unrealistic today, it may be commonplace in several years. The graph only represents the amount of memory needed initially. As mentioned above, WCSG had several severe memory leaks. So, as it routed, it would steadily increase its memory usage whether or not it needed the additional memory (unused memory would not be released back to the system). Thus, the values in the graph are actually the best case for WCSG.
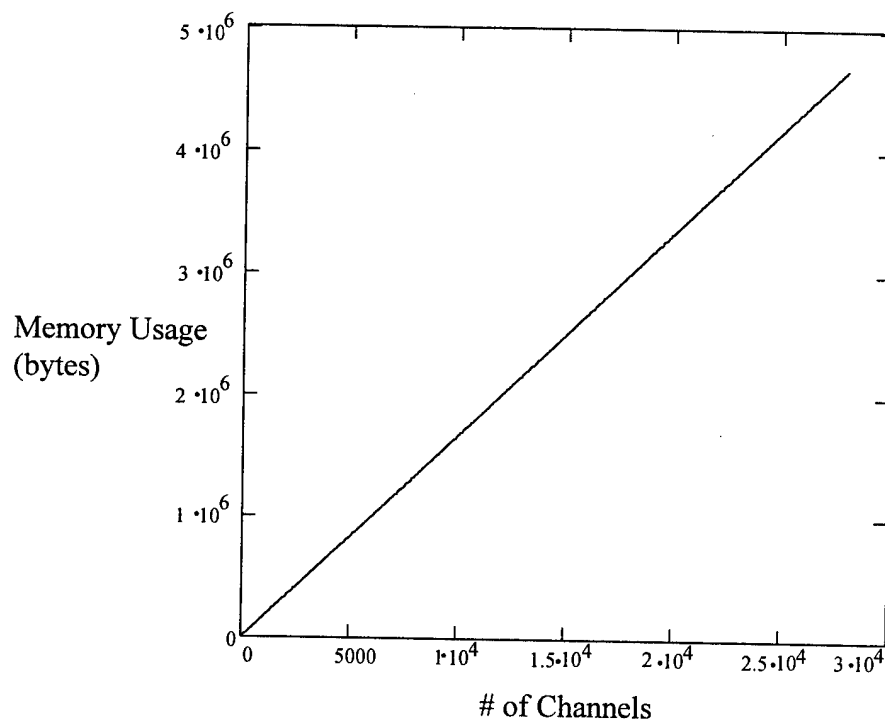
Figure 19. Current Global Router Memory Usage.

Figure 19 shows the memory usage of the global router as it currently exists. As can be seen, the memory usage is now linear and much smaller (note the y-axis values). On the high side, memory usage was reduced from 20 Gbytes to about 4 Mbyte. This is

definitely a significant difference. Practically all personal computers have at least

4 Mbytes of RAM where very few of the high-end workstations have 20 Gbytes of virtual

memory, much less 20 Gbyte of RAM.
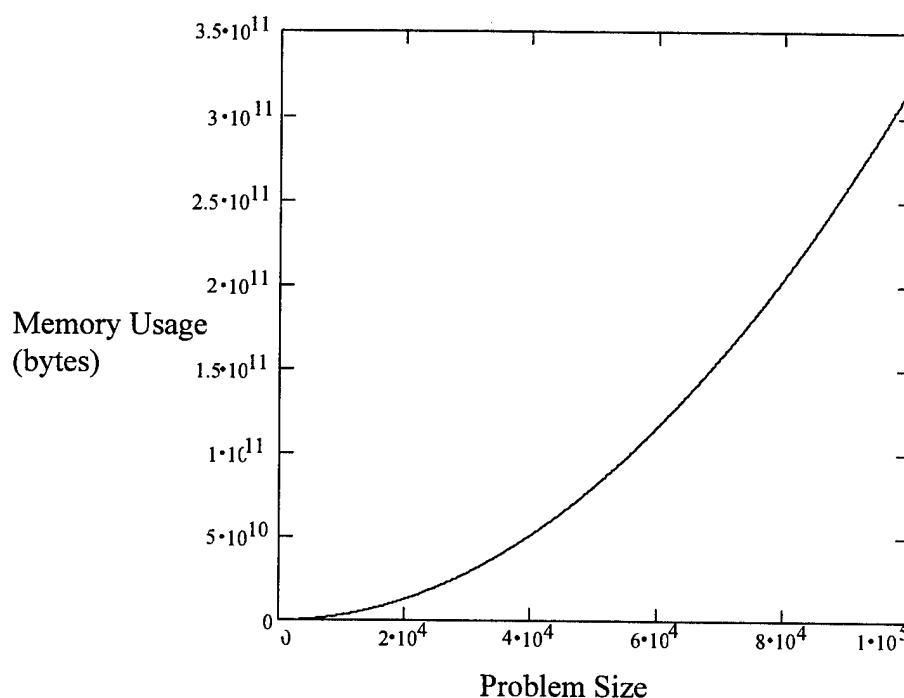
Figure 20. Old Detailed Router Memory Usage.

5.2.2 Results of Detailed Router Memory Changes

The detailed router provided by NCSU was just a simple example of a detailed router

that used Lee's algorithm. It was not meant for use on large designs. Unfortunately, in

order to use the router on a very large design (100k x 100k resolution), 320 Gbytes of

memory was needed for the grid. Figure 20 shows the memory usage of the grid in the

old detailed router versus problem size, where the problem size is the resolution of one side of a square design.

There were many code changes made to the detailed router. Of the changes made for the purpose of memory reduction, the *virtual grid* code was the most significant. The virtual grid code enables the detailed router to keep only the current channel in memory without losing data. Other channels are recreated from other, more compact, data structures as they are needed. In order to increase routing efficiency, the virtual channels are allowed to slightly overlap so that it is not necessary to blindly route a net to the edge of a channel. The overlap provides the detailed router with enough information on the next channel so that it may intelligently route the current net to an edge.

Figure 21 shows the memory usage the detailed router after the code changes were made. As in the case of the global router, the memory usage is now linear. On the high side, memory usage was reduced from 320 Gbytes to approximately 12 Mbytes. The graph assumes a near optimal channel size.

Channel size has a large affect on the memory usage with the current code. A large channel size increases the memory required to support the grid within the channel. A small channel size increases the number of channels required and therefore the overhead associated with the virtual grid is increased. As a result, there is an optimal grid size for memory usage. The optimal grid size can be approximated by:

$$M = 1.89\sqrt{y} \tag{17}$$

where M is the optimal channel size and y is the problem size (resolution of one side of a square grid). The importance of choosing a proper channel size can be seen from the

62

chart in Figure 22. This graph represents the estimated memory usage of a virtual grid with 100k x 100k resolution versus channel size.
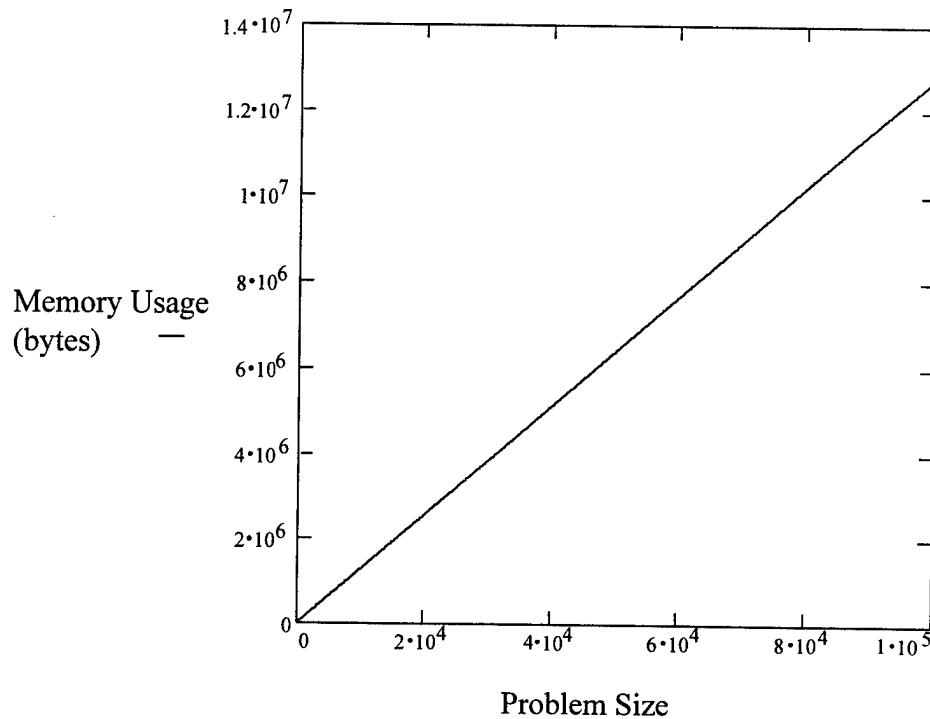
Figure 21. New Detailed Router Memory Usage.

5.2.3 Overall Memory Results

Overall, the memory usage was reduced to a reasonable amount. The router is now capable of running very large problems on a typical workstation. Overall memory usage is the sum of the usage by the global router, detailed router, channel grapher, and data structure overhead. This sum turns out to be slightly more than the memory required by the detailed router. This is because all the other components require significantly less memory than the detailed router.

63
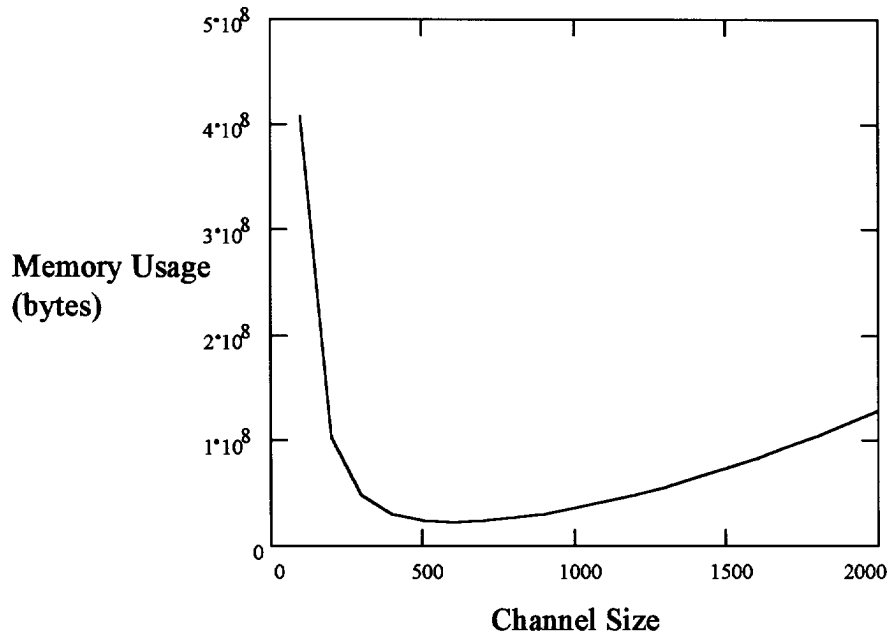
Figure 22. Memory Usage Versus Channel Size.

5.3 Results of Inertia Code

The inertia code was designed to decrease the number of bends and vias in a route. Under certain circumstances, the detailed router had a tendency to zigzag or stair-step a net. The reason this happened was because the router had a preferred direction of routing. For example in an x-y grid, the router always tried to route up first, then right, then down, and then left. So if there was a staggered set of obstacles up and to the right of the routing point, the router would route the net to follow that stagger. Additionally, any nets routed after that first net would follow the zigzag of the first net. This was a potential problem for a router that tried to minimize noise.

The inertia code was designed to alleviate this problem. The inertia code makes the preferred direction of movement the same direction in which the net was last routed. If this is the very first movement, then a calculated guess is taken as to which direction is

64

best. This was implemented such that a route can not be longer than the shortest path. Thus path length is not increased. The inertia code was also extended to include nets passing from one channel into another.

Obstacles

with
inertia code

without
inertia code
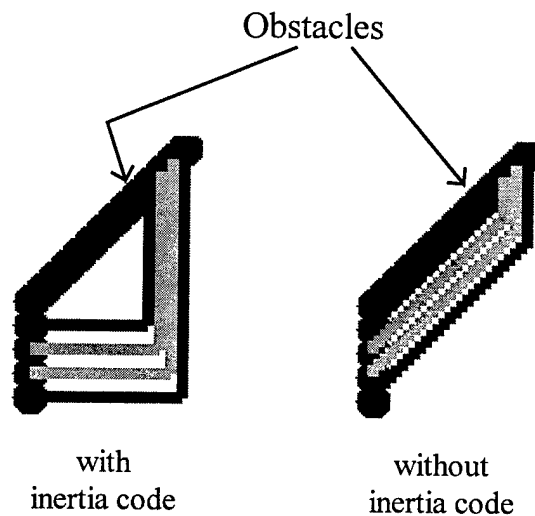
Figure 23. A Route With and Without Inertia Code.

Figure 23 shows the Xfig plot of two actual routes. The potential benefit of the inertia code becomes obvious when looking at Figure 23. It should also be pointed out that for every bend in the route, there is also a via present. So, there are four vias in the first route compared to 92 vias in the second route.

One unexpected side effect of the inertia code was the increase in routing efficiency. Table 5 lists the results of routing each design with and without the inertia code. The 25 designs listed include 20 randomly generated designs, a 4-chip and a 9-chip design from the Mayo Foundation, and two MCC designs (including a 45 μm and a 75 μm resolution version of MCC2).

Table 5. Inertia Code Comparison.

Design	# Nets	With Inertia Code				Without Inertia Code			
		# Routed	% Routed	# vias	time hh:mm	# Routed	% Routed	# vias	time hh:mm
ex1	440	263	59.8	3293	05:44	234	53.2	4804	05:38
ex2	243	182	74.9	2098	03:58	164	67.5	2321	04:06
ex3	255	192	75.3	2081	03:20	167	65.5	2449	03:07
ex4	462	261	56.9	3822	05:42	240	51.9	5804	05:47
ex5	252	188	74.6	2167	02:41	167	66.3	2991	02:58
ex6	550	309	56.2	5513	06:18	277	50.4	7987	07:48
ex7	754	298	39.5	4566	14:28	232	30.8	4977	11:20
ex8	1056	388	36.7	7172	16:17	304	28.8	7663	14:39
ex9	704	275	39.1	5051	11:10	213	30.3	5312	10:53
ex10	833	374	44.9	7075	12:12	326	39.1	8906	13:47
ex11	550	269	48.9	4178	08:57	217	39.5	4306	08:28
ex12	260	202	77.7	2455	03:43	187	71.9	2824	03:29
ex13	480	263	54.8	3820	06:40	250	52.1	5542	07:17
ex14	252	195	77.4	2342	03:08	193	76.6	3224	03:24
ex15	792	320	40.4	6018	16:25	278	35.1	7118	14:38
ex16	504	248	49.2	3287	07:48	223	44.2	4373	09:08
ex17	154	132	85.7	1150	01:46	125	81.2	1508	01:47
ex18	492	320	65.0	4928	08:02	282	57.3	7387	06:51
ex19	319	245	76.8	2818	03:02	223	69.9	4150	03:22
ex20	153	115	75.2	991	01:48	111	72.5	1270	01:55
4chip	145	145	100.0	821	00:24	144	99.3	1248	00:24
9chip	926	910	98.3	4514	02:38	886	95.7	8912	02:18
mcc1	800	531	66.4	3346	03:44	430	53.8	9826	03:47
subtotal	**11376**	**6625**	**58.2**	**83506**	**149:55**	**5873**	**51.6**	**114902**	**146:51**
mcc2-75	7119	3969	55.8	24746	39:12	3322	46.7	54417	43:56
mcc2-45	7119	4355	61.2	30050	73:52	4025	56.5	62147	77:30
Total	**25614**	**14949**	**58.4**	**138302**	**262:59**	**13215**	**51.6**	**231609**	**267:27**

Table 5 shows that the inertia code definitely increases routing efficiency and reduces

the number of vias. In every case, routing efficiency increased with the use of the inertia

code. The overall increase in routing efficiency was approximately 6.7%, which

translates into 13.1% more nets being routed. The number of vias also decreased in every case with the use of the inertia code. Even with 13.1% more nets being routed, the total number of vias was reduced by approximately 40%.

The time varied somewhat. One reason a route with the inertia code may take **more** time than the same route without the inertia code is that more routing is being done. A higher routing efficiency means more nets are routed. One reason a route with the inertia code may take **less** time than the same route without the inertia code is that every routing failure requires the router to attempt a re-route. So, the more efficient router may take less time in this case. In other words, a high routing efficiency may either increase or decrease the time it takes to route depending on the circumstances. For example, assume net A needs to be routed through ten channels. Assume the efficient router completely routes the net on its first attempt. A inefficient router may route net A through eight channels, fail, and then route it through nine channels before it fails again. In this case, the efficient router is faster. However, if the inefficient router fails in the first channel on both attempts, then the efficient router is slower.

Table 5 shows that the 4-chip design from the Mayo foundation was completely routed using the inertia code. Mayo personnel had to manually route the design because they felt that there were no routers capable routing the design and keep the crosstalk levels below their requirements. Mayo personnel worked over 80 man-hours to route the 4-chip design and it took this router only 24 minutes, a significant savings in time and money.

Overall, the inertia code was very successful. It is now an integral part of the router, and all data in this dissertation is with the inertia code except for the data noted in this sub-section.

5.4 Comparison with Other Routers

Several routers in literature have been run on the MCC benchmarks. Table 6 lists the data available for these routers and the data for the router described in this dissertation (designated "AFIT" in the table). The data for the routers in Table 6 were provided in the following literature: SLICE [54], maze router [54], V4 [53], Devaraj [24], and Miyoshi [80]. The total number of layers column in Table 6 is the total number of routing layers the router required to route 100% of the nets. This column is not applicable to the AFIT router because it is only a two layer router.

Since this router is a maze type router, it is slow compared to the newer, non-gridded routers. Maze routers must search for a path point-by-point. The newer non-gridded routers search for a path area-by area. This greatly reduces the number of required searches and greatly increases their speed. It should be noted that the AFIT router makes three passes when it routes. Each pass attempts to route any nets that were not routed in the previous pass. Therefore, the run-times for one pass would be much less than those shown in the table. It should also be noted that the pre-characterization code takes up to 40% of the total run-time of the router. So, the run-times of this router are comparable with those of the maze router in literature.

Table 6. Router Comparison.

	MCC1			MCC2-75			MCC2-45		
	time (min)	% in 2 layers	total # layers	time (min)	% in 2 layers	total # layers	time (min)	% in 2 layers	total # layers
AFIT	224	81.4	N/A	1282	62.7	N/A	8116	75.0	N/A
Maze	59		5	note 1	note 1	note 1	note 1	note 1	note 1
SLICE	12	53.1	5	445	33.9	7			
V4	4	74.3	6	60	56.7	8			
Devaraj	1.25		6	39		6			
Miyoshi	0.3		4	6.5		6	6		4

note 1: The maze router was unable to run MCC2 due to extreme memory requirements.

Other than speed, there are three important things to notice in Table 6. The first is that this router has higher routing efficiencies than those of the newer non-gridded routers. The second is that this router was able to run both MCC2-75 and its larger counterpart MCC2-45. The maze router in literature was unable to run either design due to extreme memory requirements. Finally, the other routers either do not account for crosstalk at all or they inaccurately approximate crosstalk (see the discussion in Chapter 3).

It should be noted that the channel size does effect the routing efficiency of the router. Generally, a larger channel size means a higher routing efficiency and a slower time. The channel sizes in Table 6 for MCC1, MCC2-75, and MCC2-45 were 598, 508, and 677, respectively. These values were chosen such that the designs would be evenly divided with channels. The channel sizes in Table 5 were the default of 400. Thus, the values in the two tables may vary somewhat. It is probable that the routing efficiencies for these designs could be increased even more by experimenting with different channel sizes.

69

Even though these are standard benchmarks for MCM routing, there are some inconsistencies in how they are implemented. For example, the power and ground pins can be handled in two different ways. The first way is to ignore them and assume that they are connected to reference planes so that they do not need to be routed. The second way is to route them like any other net.

The first approach to power and ground pins is the most popular because it inflates the routing efficiencies and decreases the run-times. Routing power and ground pins within the routing layers greatly reduces the routing efficiency, because they generally have a great deal more connections than the other nets. The worst case is always assumed in this dissertation; therefore, the data shown for the AFIT router includes routing the power and ground pins. It is unknown how the other routers handled the power and ground pins.

VI. Crosstalk Results

6.1 Router Crosstalk Calculations

In order for the internal crosstalk calculations to be useful, they need to be accurate. Since ContecSPICE was used as the simulator in this effort, it was also used as the standard with which to compare the router's internal crosstalk calculations.

The first test was to compare the router's crosstalk values with ContecSPICE's values for small samples. These samples were created manually to represent net configurations commonly seen in typical designs. Two of the samples are shown in Figure 24.
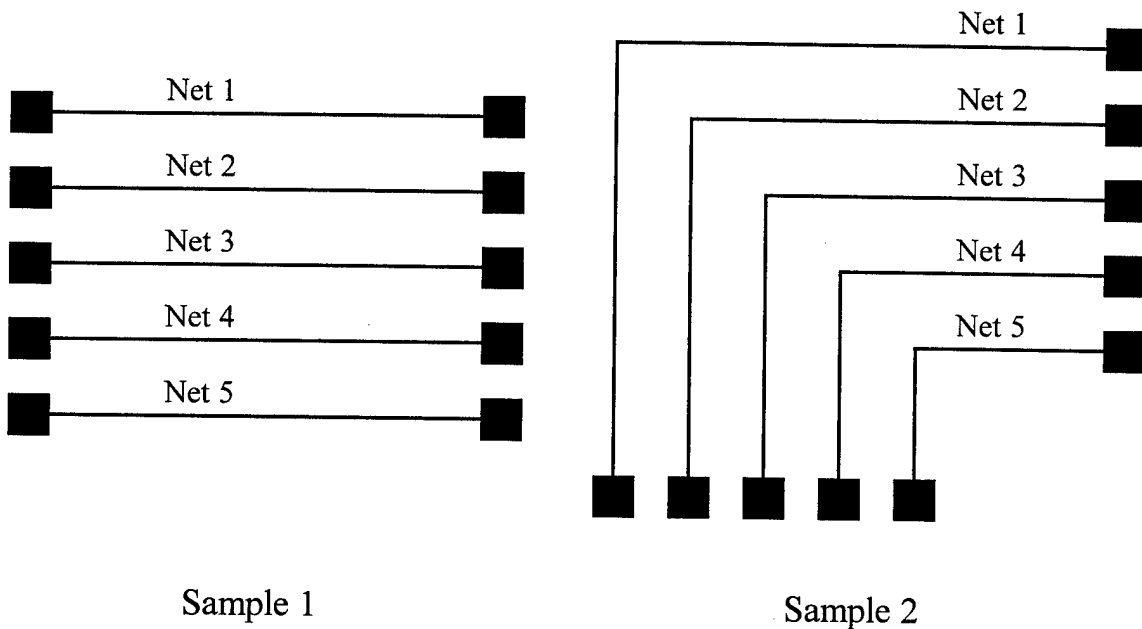
Sample 1   Sample 2

Figure 24. Samples for Crosstalk Calculations.

71

The results for sample 1 and sample 2 are in Table 7 and Table 8. Each sample was run multiple times varying which nets were present. These tables show that the rough crosstalk calculations by the router are quite accurate. They are certainly accurate enough to use a first estimate until ContecSPICE is run. In many cases, these calculations are accurate enough to stand alone. They data in these tables also show that the router generally misses on the conservative side of the true crosstalk value.

Table 7. Results from Sample 1.

	ContecSPICE	Router	% Difference
Net 1	0.04426	0.04558	3.0
Net 2	0.08323	0.08635	3.7
Net 3	0.08538	0.08892	4.1
Net 4	0.08323	0.08635	3.7
Net 5	0.04426	0.04558	3.0
	ContecSPICE	Router	% Difference
Net 1	0.00804	0.00849	5.6
Net 3	0.05007	0.05184	3.5
Net 4	0.08102	0.08266	2.0
Net 5	0.04381	0.04446	1.5
	ContecSPICE	Router	% Difference
Net 1	0.04199	0.04189	-0.2
Net 2	0.04814	0.04927	2.3
Net 4	0.04814	0.04927	2.3
Net 5	0.04199	0.04189	-0.2
	ContecSPICE	Router	% Difference
Net 1	0.00217	0.00223	2.8
Net 4	0.04241	0.04300	1.4
Net 5	0.0411	0.04077	-0.8

Table 8. Results from Sample 2.

	ContecSPICE	Router	% Difference
Net 1	0.04574	0.04649	1.6
Net 2	0.08251	0.08380	1.6
Net 3	0.07679	0.07769	1.2
Net 4	0.06626	0.06708	1.2
Net 5	0.03265	0.03297	1.0
	ContecSPICE	Router	% Difference
Net 1	0.00738	0.00801	8.5
Net 3	0.04299	0.04341	1.0
Net 4	0.06437	0.06394	-0.7
Net 5	0.03232	0.03214	-0.6
	ContecSPICE	Router	% Difference
Net 1	0.04347	0.04297	-1.2
Net 2	0.04833	0.04913	1.7
Net 4	0.03637	0.03662	0.7
Net 5	0.03100	0.03022	-2.5
	ContecSPICE	Router	% Difference
Net 1	0.00172	0.00190	10.5
Net 4	0.03159	0.03129	-0.9
Net 5	0.03040	0.02939	-3.3

Notice that the larger errors occur in sample 2 on net 1 when net 2 is not present. This is caused by the differences between calculation and simulation. When using equations to calculate the crosstalk values, the worst case must be assumed. The worst case is that all induced crosstalk pulses overlap. However, this is not always the case. Figure 25 shows one possible situation where the crosstalk pulses may not overlap. The uncoupled region of net 1 creates a temporal space between the two pulses associated with the two coupled regions. A crosstalk pulse induced on net 1 must travel through the uncoupled region while the crosstalk pulse in the second region is being induced. The gap between

the two pulses increases linearly with the length of the uncoupled region. Therefore, the crosstalk induced between net 1 and net 4 is actually less than the equations predict. This is why it is important to simulate designs. Simulation accounts for these temporal spaces caused by uncoupled regions.
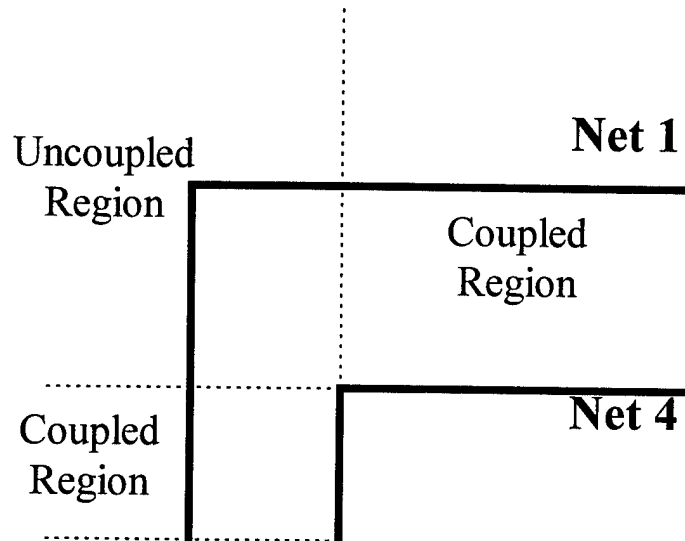
Figure 25. Uncoupled Portion of Nets with Bends.

The next step in determining the accuracy of the crosstalk prediction code was to test the code on an actual design. Figure 26 shows the crosstalk estimates for the 4-chip design versus the values from ContecSPICE. The chart shows that the estimations by the router are conservative. In some cases, the router estimates the crosstalk to be approximately double that of the ContecSPICE value. For the most part, this error is due to the irregular shapes of the nets and the effects of timing. The potential for overestimating crosstalk increases with the irregularity of the shape of the net. This is due to the timing effects discussed above.

Overall, the router's crosstalk estimations are as close to ContecSPICE's values as expected. However, the time difference more than makes up for the difference in accuracy. The router took a little over thirty minutes to route the design and calculate the crosstalk values. ContecSPICE took over five days just to calculate the crosstalk. Also, the time to calculate crosstalk by the router appears to increase linearly with the number of nets, but appears to increase exponentially in ContecSPICE. It took ContecSPICE over 45 days to calculate the crosstalk in the 9-chip design.
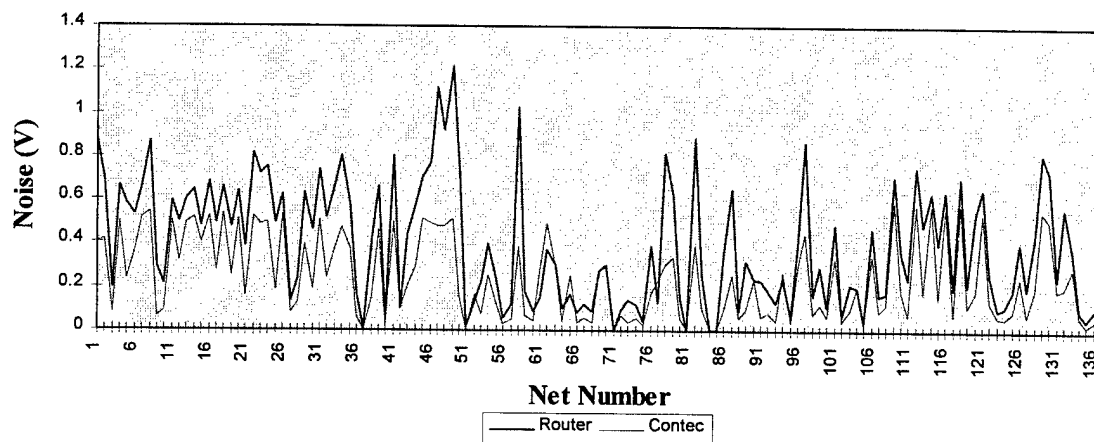
Figure 26. Router Versus Contec Noise Estimates.

6.3 Crosstalk Correction Results

After the crosstalk calculations of the router were validated to be accurate, the correction routines were tested. The correction techniques discussed earlier were implemented such that the rip-up technique could be turned off at the command line. This allowed the router to correct the noise problems as best it could but prevented it from removing nets that could not be brought within the noise budget. There may be

75

cases where it is better to leave the nets routed even though they violate the noise budget than to rip them up. This is especially true if the noise budget is conservative.

Table 9 shows the results of running the router on each design with all correction techniques off, correction techniques on except rip-up, and all correction techniques enabled (including rip-up). For each run, the noise budget was set at 0.15 V, which is approximately an order of magnitude below the typical noise margin for a 5 V design. This explains the high number of violations. The noise budget was set this low to fully test the correction capabilities of the router.

There was a significant increase in run-times when the correction techniques were used. The vast majority of the increase can be attributed to the reroute technique. The reroute technique uses the built-in maze router to attempt to find an alternate path for the net. Since the maze router is very slow compared to the newer routers, the reroute technique significantly increases run-times. It should be noted, however, that a higher noise budget (something more typical) would decrease the run-times because less nets would require crosstalk correction.

As a test of the reroute technique, the router was run on the 4-chip design with all the correction techniques enabled except rip-up and reroute (comparable to the second column in Table 9). The route took 41 minutes and routed 131 nets with 12 nets passing the noise restrictions. These numbers verify that the reroute technique was the technique that caused the significant run-times in Table 9. These numbers also bring up the question of whether or not the reroute technique is worth the extra time. The fact that 130 nets were routed with 28 nets passing the noise restrictions when reroute was used, and 131 nets were routed with only 12 nets passing the noise restrictions when reroute was

not used, implies that reroute is worth the extra time. This will be especially true for faster (non-gridded) routers that implement the crosstalk correction techniques discussed in this dissertation.

Table 9. Crosstalk Correction Results.

Design	# Nets	No Correction Techniques			All Except Rip-Up			All Correction Techniques		
		# Routed	Time hh:mm	# nets passed	# Routed	Time hh:mm	# nets passed	# Routed	Time hh:mm	# nets passed
ex1	440	265	06:08	2	235	10:40	10	99	25:01	99
ex2	243	182	03:32	4	164	05:48	6	70	05:53	70
ex3	255	215	03:05	3	158	06:05	7	71	06:24	71
ex4	462	238	06:06	5	255	10:29	7	89	10:04	89
ex5	252	177	02:46	4	147	05:25	5	59	05:20	59
ex6	550	308	05:30	4	281	10:07	6	93	09:31	93
ex7	754	295	15:04	7	216	24:28	19	110	16:40	110
ex8	1056	366	16:58	8	311	23:00	20	131	22:42	131
ex9	704	277	11:27	3	227	17:37	8	88	17:49	88
ex10	833	363	12:46	9	341	19:44	8	128	18:39	128
ex11	550	262	08:30	5	210	13:04	12	83	14:26	83
ex12	260	203	03:46	5	148	06:00	6	68	07:41	68
ex13	480	267	07:19	5	246	10:46	10	88	11:48	88
ex14	252	190	03:06	3	163	06:03	11	69	05:54	69
ex15	792	352	12:17	5	300	18:48	18	109	23:08	109
ex16	504	257	07:27	7	230	11:56	15	95	13:08	95
ex17	154	137	01:40	4	104	03:12	14	59	03:49	59
ex18	492	319	08:11	4	269	11:26	14	96	15:39	96
ex19	319	258	04:11	5	217	09:06	19	87	10:25	87
ex20	153	112	01:33	9	107	03:01	16	55	03:06	55
4chip	145	145	00:24	28	144	00:58	66	99	00:59	99
9chip	926	911	02:18	423	895	06:12	585	763	06:09	763
mcc1	800	528	03:03	9	418	05:48	38	228	06:07	228
Total	**11376**	**6627**	**147:07**	**565**	**5786**	**239:23**	**920**	**2837**	**260:22**	**2837**

Figure 27 shows a graph of the crosstalk values on each net in the 4-chip design from the Mayo Foundation. Series 1 contains the crosstalk values before any corrections were attempted. Series 2 contains the crosstalk values after the crosstalk correction techniques

(except rip-up) were applied. The following metric was used to determine the success of the corrections:

$$M = \frac{1}{N} \sum_{i=1}^{N} XT_i \qquad (18)$$

where M is the average noise per net, XT_i is the non-zero crosstalk on net I, and N is the number of nets. Applying this metric to the 4-chip design, the values 1.552 V, 1.159 V, and 0.0692 V were obtained for routing without any correction techniques, routing with all the correction techniques except rip-up, and routing with all the correction techniques including rip-up, respectively. This shows that the correction techniques were effective in reducing the overall noise in the design.
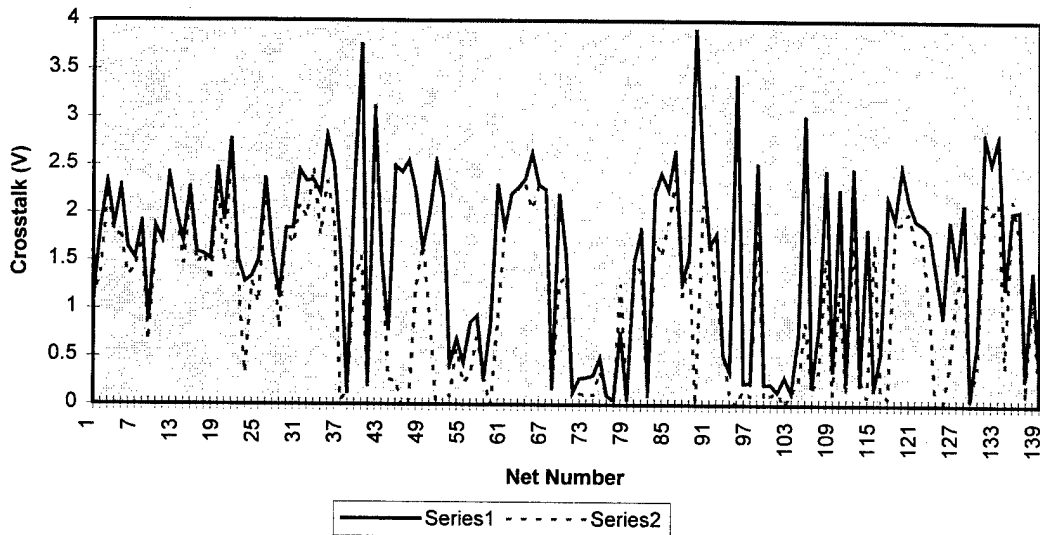
Figure 27. Noise Values Before and After Correction.

6.4 Correction Versus Avoidance

The final test for the router was to compare the approach used in this dissertation, crosstalk correction, with the typical approach in the literature, crosstalk avoidance. Typically, crosstalk avoidance is done by determining spacing rules prior to routing. The spacing rules are implemented by calculating a conservatively safe distance between nets, and then nets are not routed within that distance of each other. This method neither guarantees the avoidance of crosstalk problems, nor allows for denser routing where crosstalk problems do not exist.

The spacing rule technique of crosstalk avoidance was implemented on the router that was developed in this dissertation. This allowed for a fair comparison between that technique of crosstalk avoidance and the techniques of crosstalk correction implemented in this dissertation.

Table 10 shows the results of this comparison. The spacing rule for the designs in this table was determined to be two grid points. This was determined by running several designs with a spacing rule of one grid point and determining the number of crosstalk violations. The number of violations was extremely high in every case, so a spacing rule of two grid points was chosen. Table 10 only lists those designs that the avoidance approach created no crosstalk violations. In all nineteen designs not listed, the crosstalk avoidance approach created one or more crosstalk violations. The crosstalk correction approach did not create any violations in any design (with rip-up technique activated). The spacing rule can be increased to three grid points, but the routing efficiency would decrease even further.

Table 10. Crosstalk Avoidance Versus Crosstalk Correction.

Design	# Nets	Avoidance # Routed	Correction # Routed
ex20	153	47	55
4chip	145	90	99
9chip	926	545	763
MCC1	800	156	228
Total	**2024**	**838**	**1145**

In all four designs in which the crosstalk avoidance approach did not created crosstalk violations, the crosstalk correction approach had a higher routing efficiency. Table 10 shows that the correction approach routed 36.6% more nets, which translates into a 15.2% higher routing efficiency. This is very significant. The higher routing efficiency and the lack of any crosstalk violations in the 23 designs routed, demonstrates the importance of crosstalk correction versus crosstalk avoidance.

VII. Contributions

7.1 Novel Contributions

This dissertation makes several novel contributions to this field of study. This section lists and discusses these contributions.

The first novel contribution is in the operation of this router. This is the only known router to use online noise simulation to determine the crosstalk on each line and then use those numbers to correct the noise problems. There are several advantages to this approach. The first is that online simulation is more accurate than other methods of crosstalk calculations. The second advantage to this approach is that it is easier to modify the simulations because they are not built into the router. Finally, since crosstalk problems are corrected only after they occur (unlike other routers), maximum density can be achieved.

This is the only known router to consider *next-nearest-neighbors or next-next-nearest-neighbors* in the crosstalk calculations. Not only does this router include those potential sources of crosstalk, but this router also dynamically determines when they can be ignored. This allows the router to make trade-offs between the accuracy of its internal calculations and speed.

This is the only known router to account for the shielding of wires by wires that are in between them. All other routers calculate crosstalk pair-wise only. When three wires are on a plane, the wire in the middle shields the other two from each other. This actually

reduces the coupling and therefore the crosstalk between. Other routers us pair-wise calculations, meaning that they ignore this effect.

This is the only known router to keep a list of pointers to sources of crosstalk for each line segment. These pointers allow the router to pinpoint areas of high concentrations of noise. With these areas identified, the router can focus more time to reducing the crosstalk in these areas.

7.2 Other Contributions

This dissertation makes many other contributions to this field of study that may not be considered to be novel. This section lists and discusses these contributions.

Different capacities for each edge of each channel. The Mayo foundation expressed a problem in determining the capacity for each channel because of strangely shaped obstacles. Originally, this dissertation proposed making different capacities for the x and y directions. When implementing this, it was found that the global router code provided by NCSU would support capacities for each individual edge with very little modification.

Implementation of global pre-characterization. This appears to be the first implementation of global pre-characterization in an actual router. WCSG was created and tested by NCSU, but no detailed router had been integrated with it.

Faster global routing algorithm. The original NCSU global router code used Dikstra's algorithm to determine the shortest path between pins. Because of the assumption that all channels are square and equal sized, it was possible to greatly increase

the speed of determining the shortest path. The new algorithm is a modified Lee's algorithm (kind of a cross between Dikstra's and Lee's).

Global memory reduction. The memory usage for designs with a large number of channels was excessive. Largely due to the equal-sized square channel assumption, it was possible to greatly reduce the amount of memory used.

Virtual grid. If the grid for the detailed router is large, it can take up an excessive amount of memory also. The size of the grid is based on the resolution required and the problem size. Therefore large grids are not uncommon. Instead of keeping the entire grid in memory at once, only the active channel is in memory. The other channels are recreated from the routing data as they are needed.

Inertia code. It was found that under certain circumstances, the detailed router had a tendency to zigzag or stair-step from one point to the next. Not only does this create more bends (and increased reflected noise), but also, it causes more vias. So, some code was implemented which forces the route to continue in the same direction, if it does not make the overall route longer. A side-effect of this code was increased routing efficiency due to the reduction of the number of vias.

Coupling distance dynamically determined. Before the router begins routing, it determines how many grid spacings away need to be considered for crosstalk analysis. The number of grid spaces are different for nets on the same layer versus nets on a different layer. The number of spaces are determined using noise margins, RLGC information, and problem size.

Reflection calculation algorithm. The equation for the reflected waveform on a net with two pins is a little complicated as is shown on a simple lattice diagram. When the number of pins is increased above two, the equations become extremely cumbersome to handle. In order to handle the case of an indeterminate number of pins with an indeterminate number of reflections, a new algorithm was developed. This algorithm is based on the primitive pulse superposition idea published by NCSU.

Re-routing of overflows at the global level. Once it is determined that a net cannot be routed at the detailed level, it is sent back to the global router to chose a new path. This is not something unique to this router, but it is uncommon.

Full-matrix calculation. Other routers that route based on crosstalk issues, use pair-wise calculations between two nets. Given nets A, B, and C. The induced crosstalk on net A is not the sum of the crosstalk induced by net B and net C. This is because net C and net B interact with each other as well as with net A. Through the Contec online simulation discussed above, these interactions will be accounted for.

Crosstalk reduction code. After a problem with crosstalk has been determined, it is necessary to fix the problem. Several techniques were used to fix such problems. Very few routers attempt crosstalk correction. Most crosstalk-driven routers use crosstalk avoidance.

VIII. Conclusions and Recommendations

8.1 Conclusions

Based on the results in this dissertation, there are several conclusions that can be made. These conclusions were discussed earlier in Chapters 5 and 6 and are summarized here.

The first conclusion is that the inertia code works extremely well. In every case, the routing efficiency was increased and the number of vias was reduced by using the inertia code.

When comparing this router with others in literature, it appears that this router is slow. However, this router is a maze router and the others were not. This accounts for the vast majority of the speed differences. Also, the purpose of this dissertation was to demonstrate the crosstalk calculation and correction abilities of the router. The other routers lack these abilities.

Another conclusion is that the router's internal crosstalk calculations are fairly accurate. In many cases, the internal calculations may be accurate enough such that the online simulation may not be needed.

It also can be concluded that the crosstalk correction techniques work well. The reroute technique proved to be slow with this router, but it appears to be worth the extra time.

Finally, the techniques of crosstalk correction implemented in this dissertation are better than the spacing rule technique of crosstalk avoidance. The crosstalk correction techniques guarantee no crosstalk violations and have higher routing densities than the spacing rule technique.

8.2 Recommendations

There are still many things that can be done to improve this router. The following recommendations list potential improvements or areas of future work. The majority of this list is aimed at increasing the speed of the router. However, there are many recommendations listed that would increase other performance areas.

8.2.1 Speed

The most important change to increase the speed of the router is to replace the maze router with a non-gridded router. The maze router used in the detailed router is extremely slow compared to the newer router types. An order of magnitude decrease in the routing times is not an unrealistic goal if this is implemented.

The second most important change to increase the speed of the router is to replace the pre-characterization code in the global router. The pre-characterization code currently takes up to 40% of the overall routing time. As the code currently exists, the usefulness of the pre-characterization is very limited. Therefore, the pre-characterization code should probably be totally removed.

If the maze router is not replaced, there are a few simple improvements that would increase the speed of the maze router. Currently, there are no checks to see if two points

can be directly connected. Direct connects currently happen only after front edge of the bread crumb trail (Lee's algorithm) reaches the target point. An initial check for a direct connect can save a significant amount of time. Along those same lines, a check for a one bend or one via path to the target point may be worth implementing. Many points are connected with only one via (especially with the inertia code). An initial check would be easy to implement and may prove to be a noticeable increase in speed.

Finally, the current router has too many system calls. System calls are very time consuming. The majority of the system calls are due to memory operations. So, a new internal memory manager would be a great improvement. Also, there are many system calls that open and close various files. Many of these calls have already been reduced, but there is still room for improvement.

8.2.2 Other Performance Improvements

One item that should be changed is the use of the *realloc* command. The realloc command re-allocates a memory block to a new size, and if necessary, moves the contents from the old memory block to the new one. Unfortunately, realloc does not operate like previously thought. For example, the statement "array = (int *) malloc(sizeof(int)*1000);" allocates an array of one thousand integers. If that array needs to be increased to two thousand integers, the statement "array = (int *) realloc(array, sizeof(int)*2000);" would be used. However, it has been discovered that the original memory is not released back to the system in this example. Therefore, multiple calls to realloc wastes memory. In some cases, an extreme amount of memory can be wasted.

All uses of realloc should be removed. The command *dealloc* is a potential way to solve the problem.

Also related to memory usage are some data structures that should be modified. The *PATH* data structure should be completely removed. The need of this data structure has been overcome by events. It is no longer used. The double linked lists in the crosstalk data structures should also be removed. Single linked lists are all that are needed because the lists are very short. Also, the maintenance required for a double linked list is much higher than that for a single linked list.

Finally, the crosstalk model and equations used internal to the router could use some work. Through research or experimentation, the equations used in the code could be made more accurate and useful at a greater frequency range. It has already been shown in a previous chapter that internal crosstalk calculations are much faster than using a simulator. Increasing the accuracy of the internal calculations can eliminate the need for an external simulator in many cases.

Appendix A: SKILL Code

```
;=====================================================
;Name: _Netlist_Extract
;
;Description: This is the main function that opens the output file,
;            prints the design properties, and then prints the
;            netlist information.
;=====================================================
defun( _Netlist_Extract ()
  outputfile = axlUIPrompt( "Enter name of output file:" "router.dat" )
  if( (outputfile == nil) then exit())
  file_id = outfile( outputfile )
  fprintf( file_id "%L\n" axlDBGetProperties( axlDBGetDesign() ) )
  netlist = _Get_Netlist()
  foreach( net netlist
    _Print_Net(net file_id)
  );end-foreach
  close(file_id)
);end-defun


;=====================================================
;Name: _Get_Netlist
;
;Description: This function returns a list of dbids of all nets.
;
;=====================================================
defun( _Get_Netlist ()
  axlClearSelSet()
  axlSetFindFilter( ?enabled list( "NOALL" "NETS" ) ?onButtons list( "NETS" ))
  axlGetSelSet(axlAddSelectAll())
);end-defun


;=====================================================
;Name: _Print_Net
;
;Description: This function takes the net dbid and the port provided and
;            prints the information about the net to the port.
;
;=====================================================
defun( _Print_Net (net file_id)
  props = axlDBGetProperties( net list( "MAX_VIA_COUNT" "MIN_NOISE_MARGIN"))
  fprintf( file_id "net - %s : %d %L\n" net->name net->nBranches props)
  foreach( branch net->branches
    _Print_Branch_Info(branch file_id)
  );end-foreach
);end-defun


;=====================================================
;Name: _Print_Branch_Info
;
;Description: This function prints the information about the branch dbid
;            provided to port file_id.  This function prints the information
;            by calling the appropriate function based on the object type.
;
;=====================================================
defun( _Print_Branch_Info ( branch file_id)
  foreach( child branch->children
    case( child->objType
```

```
                    ( "pin"   _Print_Pin( child file_id))
                    ( "via"   _Print_Via( child file_id))
                    ( "path"  _Print_Path( child file_id))
                    ( "shape" _Print_Shape( child file_id))
                    ( "tee"   _Print_Tee( child file_id))
                    ( t      error( "Unknown object type: *%s* in net: *%s*\n" child->objType child->net->name))
          );end-caseq
       );end-foreach
);end-defun


;==========================================================================
;Name: _Print_Pin
;
;Description: This function prints location, starting layer, and ending layer
;             of the pin dbid to the port file_id.
;
;==========================================================================
defun( _Print_Pin ( pin file_id)
   fprintf( file_id "\tpin %P %L\n" pin->xy pin->startEnd)
);end-defun


;==========================================================================
;Name: _Print_Via
;
;Description: This function prints location, starting layer, and ending layer
;             of the pin dbid to the port file_id.
;
;==========================================================================
defun( _Print_Via ( via file_id)
   fprintf( file_id "\tvia %P %L %s\n" via->xy via->startEnd via->name)
);end-defun


;==========================================================================
;Name: _Print_Tee
;
;Description: This function prints the location and layer of the tee dbid
;             to the port file_id.
;
;==========================================================================
defun( _Print_Tee ( tee file_id)
   fprintf( file_id "\ttee %P %s\n" tee->xy tee->layer)
);end-defun


;==========================================================================
;Name: _Print_Path
;
;Description: This function prints information on the path dbid to the port
;             file_id.  The information printed includes the layer, start and
;             end points of the lines, and the width of the lines.
;
;==========================================================================
defun( _Print_Path ( path file_id)
   if( ( path->hasArcs == t) then error( "Path in net *%s* has arcs!" path->net->name))
   fprintf( file_id "\tpath - %s\n" path->layer )
   foreach( segment path->segments
     fprintf( file_id "\t\tline %L %f\n" segment->startEnd segment->width)
   );end-foreach
);end-defun


;==========================================================================
;Name: _Print_Shape
;
;Description: This function prints the layer and the information about the
;             lines outlining the shape.
;
```

```
;━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
defun( _Print_Shape ( shape file_id)
  fprintf( file_id "\tshape - %s\n" shape->layer )
  foreach( segment shape->segments
    if( ( segment->objType == "arc") then error( "Shape in net *%s* has arcs!" shape->net->name))
      fprintf( file_id "\t\tline %L %f\n" segment->startEnd segment->width)
  );end-foreach
);end-defun


;━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
;Name: _Netlist_Input
;
;Description: This function is the main function for the procedures that
;             read the routing information from infile and creates the
;             necessary vias and etches.
;━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
defun( _Netlist_Input ( )
  infile = axlUIPrompt( "Enter name of output file:" "router.dat" )
  if( ( infile == nil) then exit( ))
  file_id = infile( infile )
  when( file_id
    not_done = fscanf( file_id "%s" word)
    path_flag = t
    while( (not_done == 1)
      case( word
        ( "net" fscanf( file_id " - %s" netname) gets( eol file_id))
        ( "via" _Create_Via( file_id netname))
        ( "path" word = _Create_Path( file_id netname) ( path_flag = nil))
        ( t gets( eol file_id))
      );end-case
      if( ( path_flag == t ) then ( not_done = fscanf( file_id "%s" word)))
      if( ( path_flag == nil) then not_done = 1)
      path_flag = t
    );end-while
  );end-when
);end-defun


;━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
;Name: _Create_Via
;
;Description: This function reads in the location and padstack name from
;             the port file_id and then creates a via.
;
;━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
defun( _Create_Via ( file_id netname)
  fscanf( file_id " %f:%f %s %s %s" x y temp1 temp2 padstackname)
  temp = axlDBCreateVia( padstackname list( x y) )
  if( (temp == nil) then error( "temp in createvia is nil"))
);end-defun
;━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
;Name: _Create_Path
;
;Description: This function creates a path by reading in the line information
;             from port file_id.
;
;━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
defun( _Create_Path ( file_id netname)
  fscanf( file_id " - %s" layer)
  fscanf( file_id "\tline ((%f %f) (%f %f)) %f" x1 y1 x2 y2 width)
  path = axlPathStart( list( x1:y1 x2:y2) width)
  not_done = fscanf( file_id "%s" word)
  loop = nil
  if( (word == "line") then ( loop = t))
  while( (loop == t)
    fscanf( file_id " ((%f %f) (%f %f)) %f" x1 y1 x2 y2 width)
```

91

```
        temp = axlPathLine( path width list( x2 y2))
        if( (temp == nil) then error( "temp in pathline is nil" ))
        not_done = fscanf( file_id "%s" word)
        loop = nil
        if( ( word == "line") then ( loop = t))
    );end-while
    temp = axlDBCreatePath( path layer netname)
    if( (temp == nil) then error( "temp in createpath is nil" ))
    word
);end-defun
```

Appendix B: Router Source Code

Due to the size of the program code, it has been included into a separate volume. A

copy of the code may be obtained through:

Lt Col Tom S. Wailes, USAF
Air Force Institute of Technology
AFIT/ENG
Wright-Patterson AFB, OH 45433-7765
(513) 255-5276 x4716

Appendix C: User's Manual

C.1 Introduction

This appendix contains the user's manual for the routing system created in this dissertation. The two SKILL programs and the router are described in terms of their operation and use. The associated files of each program are also described.

C.2 SKILL Programs

The SKILL programs are used to convert between the format used by the router and the format used by Allegro. These programs, like all SKILL programs, are run from within Allegro. In order to run the programs, Allegro must be in the SKILL mode. Allegro is put in the SKILL mode by typing "skill" at the prompt.

There are two separate programs used to convert between the two formats. Each program is in a separate file. These files are called *net_in.skl* and *net_out.skl*. The best way to load these files into Allegro is to load them from the file *allegro.ilinit* with a line such as: *load("net_in.skl")*.

The program used to convert Allegro data into the format used by the router is in the file *net_out.skl*. From the SKILL command-line prompt type *_Netlist_Extract()* to activate the program. A window will appear and ask for the name of the output file. The name *router.dat* appears in the window by default, but it can be changed. After typing the name, press **enter** or click on the **done** button. The rest is done automatically. All

pins, vias, and etches are extracted and written to the output file. If something inconsistent with a Manhattan geometry is encountered, such as an arc, an error message is displayed and the program terminates.

The program used to read the router data into an Allegro design is in the file *net_in.skl*. From the SKILL command-line prompt type *_Netlist_Input()* to activate the program. As in the previously discussed program, a window will appear and ask for the name of the input file. The name *router.dat* appears in the window by default, but it also can be changed. After entering the name of the input file, the program will read the file and add the etches and vias from the file to the active design. As the file is read, the new etches and vias appear on the screen and are assigned to the appropriate net. This process is finished when the SKILL command-line prompt returns.

C.3 Router

The router is the nucleus of the system. The SKILL programs are used to support the router. The following discusses two main parts of the router: supporting files and command-line options.

C.3.1 Supporting Files

There are three important files associated with the router (other than the input and output files). Two of these files need to be provided by the user, and the other is created by the router. There are also many other files created in the interface with Contec, but these are transparent to the user except to note that certain file names are reserved.

The first file the user needs to provide is the technology file. The technology file tells the router information on each layer in the design. The technology file can usually be reused for each design that uses the same MCM technology. This file is very similar to the layer file that is created by ContecLIF when the design is extracted. Editing this layer file is probably the easiest way to create the technology file.

The technology file format is fairly simple. The first two lines are ignored in order to be consistent with the layer file created by ContecLIF. Each additional line represents a layer of the substrate going from top to bottom.

There are twelve fields on each line. Many of these fields are included to comply with the layer file format from ContecLIF. In fact, no fields are changed, but two fields are added to the ContecLIF layer file. The first field is always **S**. The next field is the layer number. Layers are numbered starting with **0** (zero) and must be in order. The third field is the layer name if it has one. The routing layers must have names. Field four is ignored. Field five is **YES** if the layer is a conductor layer or **NO** if it is not. Field six is the relative dielectric constant for the layer. Field seven is the electrical conductivity of the layer in mho/cm. Field eight is the layer material. Field nine is **YES** if the layer is a shield layer or **NO** if it is not. Field ten is the layer thickness in mils. Field eleven is **YES** if the layer is a routing layer and **NO** if it is not. Finally, field twelve is the conductivity of the dielectric material in the layer.

Each field must be separated with an exclamation point with no spaces. Figure 28 shows a sample technology file. It is important to remember that two and only two routing layers must be present.

```
A!LAYER_SORT!LAYER_SUBCLASS!LAYER_ARTWORK!...
J!./filename!date!
S!0!!!NO!1.00!0 mho/cm!AIR!!0 mil!NO!0.0!
S!1!TOP!POSITIVE!YES!11.7!595900 mho/cm!COPPER!NO!0.197 mil!NO!0.001!
S!2!IOPADS!POSITIVE!YES!11.7!595900 mho/cm!COPPER!NO!0.197 mil!NO!0.001!
S!3!!!NO!11.7!0 mho/cm!POLYMER!!0.098 mil!NO!0.001!
S!4!S2!POSITIVE!YES!11.7!595900 mho/cm!COPPER!NO!0.197 mil!YES!0.001!
S!5!!!NO!11.7!0 mho/cm!POLYMER!!0.098 mil!NO!0.001!
S!6!S1!POSITIVE!YES!11.7!595900 mho/cm!COPPER!NO!0.197 mil!YES!0.001!
S!7!!!NO!1.00!0 mho/cm!AIR!!0 mil!NO!0.0!
```

Figure 28. Sample Technology File.

The second file that the user must provide is the configuration file. The configuration file contains information about the particular design. The router will prompt for the information in the configuration file if it is omitted. However, it much more convenient to include the configuration file.

The format of the configuration file is to have the field name (case insensitive) followed by a single space and then the value. The final line of the file should be an asterisk. The allowable field names are: gamma, pinimpedance, xmax, xmin, ymax, ymin, stepsize, frequency, viapadstack, pinpadstack, datapath, linewidth, Vin, noisemargingood, and noisemarginreject.

Figure **29** shows a sample configuration file.

Gamma is the reflection coefficient for net terminals. Pinimpedance is the input impedance of each net terminal. The fields, pinimpedance and gamma, should not both be used. Xmax, xmin, ymax, and ymin define the boundaries of the design. Stepsize

defines the resolution of the grid. Linewidth defines the width of each etch. Linewidth

must be less than the stepsize. Vin is the input voltage in volts. Noisemargingood is the

noise level in volts that if any nets exceed , the router will warn the user.

Noisemarginreject is the noise level in volts that if any nets exceed, the router will correct

the net. Frequency is in hertz. Viapadstack is the name of the padstack of the vias in

Allegro. Pinpadstack is the name of padstack of the pins in Allegro. Finally, datapath is

the directory in which to look for the pre-characterization data if it is not in the current

directory.

```
pinimpedance 50
xmax 45000
xmin 0
ymax 45000
ymin 0
stepsize 50
linewidth 32.0
vin 5.0
noisemargingood 1.0
noisemarginreject 1.4
frequency 1e9
viapadstack VIA_S2_S1
pinpadstack SMD132_75_UB
datapath /students/kjmcclel/diss/data
*
```

Figure 29. Sample Configuration File.

The other important file that needs to be noted is the *graph.fig* file. This file is an Xfig compatible file that graphically represents the results of the routing. Pins are marked as red dots and the etches are color-coded for each net.

There are quite a few file names that are reserved for the interface with Contec. The simplest way to avoid any problems is have no files that start with *contec* in their name. For example, the file *contec_rlgc.in* would be overwritten if it was in the current directory.

C.3.2 Command-Line Options

There are many different command-line options available with the router. The command-line should look like: *router [-c configfile] [-t techfile] [-o outfile] [-g N] [-l N] [-i] [-n] [-r] [-s] [-d] [-x] [-u] infile*. The following discusses each command-line option. It should be noted that the command-line options are not case sensitive and are not order sensitive.

The *infile* term on the command-line is the only term that is required. This is the name of the input file and may or may not include a *.dat* extension. For example, if the input file was *4chip.dat*, then either *4chip* or *4chip.dat* would work for the *infile* term.

The *[-c configfile]* option changes the name of the configuration file. If this option is not present, the filename defaults to *default.cfg*.

The *[-t techfile]* option changes the name of the technology file. If this option is not present, the filename defaults to *default.tch*.

The *[-o outfile]* option changes the name of the output file. If this option is not present, the filename defaults to *infile.out*. In this case, *infile* is the name of the input file without the *.dat* extension.

The *[-g N]* option changes the size of the virtual grid (channel size). The default size is 400. For example, to change the channel size to 550, the following line would work: *router -g 550 infile.*

The *[-l N]* option changes the number of loops the router makes through the entire routing process. The default is two. For example, to limit the router to one attempt to globally route and then to detail route a design, the following line would be appropriate: *router -l 1 infile.*

The *[-i] [-n] [-r] [-s] [-d] [-x] [-u]* options turns off the inertia code, turns off all crosstalk calculations, turns off the reflection calculations, turns off the on-line simulation, allows on-line simulation for the entire design only (not for each channel), turns off all crosstalk correction code, and turns off the rip-up correction technique; respectively. Obviously, some options supersede others. For example, in the line *router -n -s -x infile*, the *-s* and *-x* options are ignored because the *-n* option turned off all crosstalk calculations.

Bibliography

1. Alaybeyi, M.M., et al. "Analysis of MCMs using Asymptotic Waveform Evaluation (AWE)," Proceedings 1992 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1992.

2. Antinone, Robert J. and Gerald W. Brown. "The Modeling of Resistive Interconnects for Integrated Circuits." In Stuart K. Tewksbury, editor. Microelectronics System Interconnections: Performance and Modeling. New York: IEEE Press, 1994.

3. Bakoglu, Halil. Circuit and System Performance Limits on ULSI: Interconnections and Packaging. Ph.D. Dissertation, Stanford University. October 1986.

4. Bakoglu, H.B. and James D. Meindl. "Optimal Interconnection Circuits for VLSI." In Stuart K. Tewksbury, editor. Microelectronics System Interconnections: Performance and Modeling. New York: IEEE Press, 1994.

5. Becker, Wiren D. "FDTD Modeling of Noise in Computer Packages," Proceedings 1993 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1993.

6. Belkerdid, M.A. and P.F. Wahid. "Rise Time and Frequency Analysis for Crosstalk in High-Speed Packaging," Proceedings 8th Annual International Electronics Packaging Conference. Wheaton: International Electronics Packaging Society Inc., 1988.

7. Bose, Subroto and Bidyut K. Bhattacharyya." Time Domain Method for Optimizing Transmission Line Model," Proceedings 8th Annual International Electronics Packaging Conference. Wheaton: International Electronics Packaging Society Inc., 1988.

8. Brews, John R. "Transmission Line Models for Lossy Waveguide Interconnections in VLSI." In Stuart K. Tewksbury, editor. Microelectronics System Interconnections: Performance and Modeling. New York: IEEE Press, 1994.

9. Buchanan, J. E. BiCMOS/CMOS Systems Design. McGraw Hill, 1990.

10. Bumalough, Anthony B. "Modeling of Reflections Caused by Discontinuities in Printed Wiring Boards," Proceedings 8th Annual International Electronics Packaging Conference. Wheaton: International Electronics Packaging Society Inc., 1988.

11. Cangellaris, A.C., et al. "Real Inductance Calculation for High-Speed Interconnect Systems," Proceedings 8th Annual International Electronics Packaging Conference. Wheaton: International Electronics Packaging Society Inc., 1988.

12. Canright, Robert E., Jr. "Equations for Crosstalk and Controlled Impedance," Proceedings 8th Annual International Electronics Packaging Conference. Wheaton: International Electronics Packaging Society Inc., 1988.

13. Caverly, Robert H. "Characteristic Impedance of Integrated Circuit Bond Wires." In Stuart K. Tewksbury, editor. Microelectronics System Interconnections: Performance and Modeling. New York: IEEE Press, 1994.

14. Chambers, Ben, et al. "Evaluation of a Multi-Chip Module (MCM) Thermal Design Tool- A Three Chip MCM Thermal Mock-Up as a Case Study," ICEMM Proceedings, 1993.

15. Chang, C.S. "Transmission Lines." In A.E. Ruehli, editor. Circuit Analysis, Simulation and Design, volume 3, part 2 of Advances in CAD for VLSI. chapter 11.3. Elsevier Science Publishers, 1987.

16. Chang, Fung-Yuel. "Transient Simulation of Frequency-dependent Nonuniform Coupled Lossy Transmission Lines with Double Orthogonal Expansions and Recursive Convolution Integrations," Proceedings 1993 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1993.

17. Chen, Howard H. and C. K. Wong. "63-layer TCM Wiring with Three-Dimensional Crosstalk Constraints." Manuscript awaiting publication.

18. Chen, Yuzhe, et al. "Modeling of Delta-I Noise In Digital Electronics Packaging," Proceedings 1994 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1994.

19. Chiprout, Eli and Michel Nakhla. "Transient Waveform Estimation of High-Speed MCM Networks Using Complex Frequency Hopping," Proceedings 1993 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1993.

20. Cohen, Paul D. "Waveform Simulation in High Speed PCB Design," Proceedings 8th Annual International Electronics Packaging Conference. Wheaton: International Electronics Packaging Society Inc., 1988.

21. Dai, Wayne W., et al. "Cost-Driven Layout for Thin-film MCMs," <u>Proceedings 1993 IEEE Multi-Chip Module Conference</u>. Washington: IEEE Computer Society Press, 1993.

22. Dai, Wayne W. "Multichip Routing and Placement," <u>IEEE Spectrum</u>. **29**:61-64. New York: IEEE Press, November 1992.

23. Davidson, E.E. and G.A. Katopis. "Package Electrical Design." In R. Tummala and E. Rymaszewski, editors. <u>Microelectronics Packaging Handbook</u>, chapter 3. New York: Van Nostrand Reinhold, 1989.

24. Devaraj, Giriaj and Dinesh Bhatio. "Crosstalk Driven MCM Router," <u>Journal of Microelectronic Systems Integration</u>. **2**:65-80. Plenum Publishing, 1994.

25. Djordjevic, Antonije R. et al. "Time-Domain Response of Multiconductor Transmission Lines." In Stuart K. Tewksbury, editor. <u>Microelectronics System Interconnections: Performance and Modeling</u>. New York: IEEE Press, 1994.

26. Doane, Daryl A. "Materials and Fabrication Techniques for Packages and Circuit Boards at Microwave Frequencies: An Overview," <u>Proceedings 8th Annual International Electronics Packaging Conference</u>. Wheaton: International Electronics Packaging Society Inc., 1988.

27. Donlin, Mike, editor. "Packaging Innovations Help Engineers Break Free From Design Constraints," <u>Computer Design</u>. **32**:65+. Westford: Pennwell, June 1993.

28. Fan, J., et al. "Physical Layout Algorithms for Computer Generated Holograms in Optoelectric MCM Systems Design," <u>Proceedings 1993 IEEE Multi-Chip Module Conference</u>. Washington: IEEE Computer Society Press, 1993.

29. Feller, A., et al. "Crosstalk ad Reflections in High-Speed Digital Systems," <u>AFIPS Conference Proceedings - Fall Joint Computer Conference</u>, 1965.

30. Franzon, Paul, et al. "Automatic A-Priori Generation of Delay and Noise Macromodels and Wiring Rules for MCMs," <u>Proceedings 1993 IEEE Multi-Chip Module Conference</u>. Washington: IEEE Computer Society Press, 1993.

31. Franzon, Paul, et al. "System Design Optimization for MCM," <u>Technical Report NCSU-ERL-94-16</u>. North Carolina State University, 1994.

32. Franzon, Paul, et al. "Tools to Aid in Wiring Rule Generation for High Speed Interconnects," <u>Proceedings of the Design Automation Conference</u>, 1992.

33. Franzon, Paul. Notes for IC702P. North Carolina State University. January 1994.

34. Franzon, Paul. Professor of Electrical Engineering, North Carolina State University, Raleigh NC. Personal interview. 19-20 December 1994.

35. Gao, David S., et al. "Modeling and Simulation of Interconnection Delays and Crosstalks in High-Speed Integrated Circuits." In Stuart K. Tewksbury, editor. Microelectronics System Interconnections: Performance and Modeling. New York: IEEE Press, 1994.

36. Gilbert, Barry K. Gigabit System Design, Packaging, and Interconnect. Final Report for 02/25/87 - 12/31/90. Mayo Foundation, 1992

37. Guruswamy, Mohan and D. F. Wong. Echelon: A Multilayer Detailed Area Router," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 15:1126-1136. New York: IEEE Press, September 1996.

38. Hakim, Ed. "1993 International Conference on MCMs," Solid State Technology. 36:99-100. Westford: Pennwell, July 1993.

39. Harvatis, Christoforos, et al. "Topological Description and Interconnect Delay Modeling for Performance-Driven Partitioning and Placement of MCM Design," Technical Report NCSU-ERL-94-17. North Carolina State University, 1994.

40. Hayes, Paul R. Electrical Engineer, Special Purpose Processor Development Group, Mayo Foundation, Rochester Mn. Personal interview. 1 February 1995.

41. Healy, Steven T. "An Improved Model for Solving the Optimal Placement for River-Routing Problem," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 12:1473-1480. New York: IEEE Press, October 1993.

42. Hwang, Lih-Tyng and Arnold Reisman. "The Effects of the Skin Depth of a Thin-Film Package," Proceedings 8th Annual International Electronics Packaging Conference. Wheaton: International Electronics Packaging Society Inc., 1988.

43. Iqbal, Asif, et al. "Design Tradeoffs Among MCM-C, MCM-D and MCM-D/C Technologies," Proceedings 1993 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1993.

44. Iqbal, Asif. "Full Wave Electrical Modeling of MCM by Robust Design Methodology," Proceedings 1992 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1992.

45. Jhang, Kyoung-Son, et al. "COP: A Crosstalk Optimizer for Gridded Channel Routing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. **15**:424-429. New York: IEEE Press, April 1996.

46. Johnson, Howard W. and Martin Graham. High-Speed Digital Design: A Handbook of Black Magic. Englewood Cliffs: Prentice Hall, 1993.

47. Joobbani, Rostam. An Artificial Intelligence Approach to VLSI Routing. Hingham: Kluwer Academic Press, 1986.

48. Kalafala, A.K., et al. "CC3D - A Three-Dimensional Electromagnetic Design/Analysis Code for Electronic Packaging," Proceedings 8th Annual International Electronics Packaging Conference. Wheaton: International Electronics Packaging Society Inc., 1988.

49. Katopis, G. A. and H. H. Smith. "Coupled Noise Predictors for Lossy Interconnects," CPMT. December 1994.

50. Kayssi, Ayman and Karem A. Sakallah. "Delay Macromodels for Point-to-Point MCM Interconnections," Proceedings 1992 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1992.

51. Kear, Fred W. Hybrid Assemblies and Multichip Modules. New York: Marcel Dekker, Inc, 1993.

52. Kennedy, Jeff. "Measuring and Modeling High-Speed IC Packaging Effects," Computer Design. **32**:69+. Westford: Pennwell, June 1993.

53. Khoo, Kei-Yong and Jason Cong. "A Fast Four-Via Multilayer MCM Router," Proceedings 1993 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1993.

54. Khoo, Kei-Yong and Jason Cong. "A Fast Multilayer Area Router for MCM Designs," IEEE Transactions on Circuits and Systems II. **39**:841-851. New York: IEEE Press, November 1992.

55. Kim, Jaeseok and John F. McDonald. "Transient and Crosstalk Analysis of Slightly Lossy Interconnection Lines for Wafer Scale Integration and Wafer Scale Hybrid Packaging - Weak Coupling Case." In Stuart K. Tewksbury, editor. Microelectronics System Interconnections: Performance and Modeling. New York: IEEE Press, 1994.

56. Kirkpatrick, D. A. and A. L. Sangiovanni-Vincetelli. "Techniques for Crosstalk Avoidance in the Physical Design of High-Performance Digital Systems," University of California/Berkeley for Semiconductor Research Corporation Publication C94609. North Carolina, December 1994.

57. Kuo, Sy-Yen. "YOR: A Yield-Optimizing Algorithm by Minimizing Critical Areas and Vias," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 12:1303-1311. New York: IEEE Press, September 1993.

58. Lall, Pradeep and Shikar Bhagath. "An Overview of Multichip Modules," Solid State Technology. 36:65-71. Westford: Pennwell, September 1993.

59. Le Coz, Y.L. and R.B. Iverson. "A High-Speed Multi-Dielectric Capacitance-Extraction Algorithm for MCM Interconnects," Proceedings 1992 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1992.

60. Lee, C. "An Algorithm for Path Connections and Its Applications," IRE Transactions on Electronic Computing. 346-365. September 1961.

61. Lee, Jaebum and Eugene Shragowitz. "Synthesis of Transmission Line Interconnects in Presence Distortions," Technical Report TR 94-39. Computer Science Department, University of Minnesota, July 1994.

62. Lewis, E.T. "An Analysis of Interconnect Line Capacitance and Coupling for VLSI Circuits." In Stuart K. Tewksbury, editor. Microelectronics System Interconnections: Performance and Modeling. New York: IEEE Press, 1994.

63. Liao, Haifang and Wayne Wei-Ming Dai. "Wave Spreading Evaluation of Interconnect Systems," Proceedings 1993 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1993.

64. Liao, J.C. and G.N. Choksi. "An Integrated Approach for Electrical Performance Analysis of Multichip Modules," Proceedings 1992 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1992.

65. Lin, Shen and Ernest S. Kuh. "Pade Approximation Applied to Transient Simulation of Lossy Coupled Transmission Lines," Proceedings 1992 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1992.

66. Lindsted, Robert D. and Rohinton J. Surty. "Steady-State Junction Temperatures of Semiconductor Chips," IEEE Transactions on Electronic Devices. 19:41-44. New York: IEEE Press, January 1972.

67. Luo, Sifen, et al. "Crosstalk in Coupled Interconnects with Meshed Ground Planes," <u>Proceedings 1994 IEEE Multi-Chip Module Conference</u>. Washington: IEEE Computer Society Press, 1994.

68. Maitre, Emmanuel A. <u>Crosstalk Evaluation Using Closed-Form Expressions</u>. Project Report for Dr. Paul Franzon. North Carolina State University, December 1994.

69. Maley, F. Miller. <u>Single-Layer Wire Routing and Compaction</u>. Cambridge: MIT Press, 1989.

70. Maliniak, David. "Ceramic-Substrate Technology Creates Reworkable MCMs," <u>Electronic Design</u>. **41**:35-36. Cleveland: Penton Publishing, April 1993.

71. Maliniak, David, editor. "Low-Cost Developments Surface for Multichip-Module Technology," <u>Electronic Design</u>. **40**:28. Cleveland: Penton Publishing, November 1992.

72. Maliniak, David, editor. "MCMs: 11 Experts Debate the Future," <u>Electronic Design</u>. **40**:157+. Cleveland: Penton Publishing, November 1992.

73. Maliniak, David. "Infrastructure Issues Restrain MCMs," <u>Electronic Design</u>. **40**:28. Cleveland: Penton Publishing, November 1992.

74. Maliniak, David. "MCM Materials Gains Have Diamonds Glitter," <u>Electronic Design</u>. **41**:36+. Cleveland: Penton Publishing, April 1993.

75. Maliniak, Lisa. "Interconnect Synthesis Uses Electrical Descriptions to Create High-Speed PC Boards and MCMs That Work," <u>Electronic Design</u>. **42**. Cleveland: Penton Publishing, May 30, 1994.

76. Matthaei, G.L. "Crosstalk." In S.I. Long and S.E. Butner, editors. <u>Gallium Arsenide Digital Integrated Circuit Design</u>. chapter 5.5. New York: McGraw-Hill, Inc., 1990.

77. Mazzullo, Tony. "The Future of Packaging Design," <u>Electronic Design</u>. **40**:120. Cleveland: Penton Publishing, November 1992.

78. Mehrotra, Sharad. <u>Automated Synthesis of High Speed Digital Circuits and Package-Level Interconnect</u>. Ph.D. Dissertation. North Carolina State University, 1994.

79. Mentor Graphics. "Foundry-Specific MCM Design Kit," <u>Electronic Design</u>. **41**:S22+. Cleveland: Penton Publishing, September 1993.

80. Miyoshi, Tetsuya, et al. "An MCM Routing Algorithm Considering Crosstalk," IEEE Press, 1995.

81. Mowatt, Larry. "Larry Mowatt on: Multichip Modules," <u>Computer design</u>. **32**:115-117. Westford: Pennwell, July 1993.

82. Nakhla, Michel and Qi-jun Zhang. "Simulation and Optimization of Interconnect Delay and Crosstalk in Multi-Chip Modules," <u>Proceedings 1992 IEEE Multi-Chip Module Conference</u>. Washington: IEEE Computer Society Press, 1992.

83. Nayak, Deepak, et al. "Calculation of Capacitances of Chip-to-Chip Interconnections on a High Density Multi-Chip Package," <u>Proceedings 8th Annual International Electronics Packaging Conference</u>. Wheaton: International Electronics Packaging Society Inc., 1988.

84. Palusinski, Olgierd A. and Anyu Lee. "Analysis of Transients in Nonuniform and Uniform Multiconductor Transmission Lines." In Stuart K. Tewksbury, editor. <u>Microelectronics System Interconnections: Performance and Modeling</u>. New York: IEEE Press, 1994.

85. Palusinski, Olgierd A., et al. "Electrical Modeling of Interconnections in Multilayer Packaging Structures." In Stuart K. Tewksbury, editor. <u>Microelectronics System Interconnections: Performance and Modeling</u>. New York: IEEE Press, 1994.

86. Pan, G., et al. "On the Study of Skin-Effect and Dispersion of Heavily Lossy Transmission Lines," <u>Proceedings 1992 IEEE Multi-Chip Module Conference</u>. Washington: IEEE Computer Society Press, 1992.

87. Pect, Michael. <u>Integrated Circuit, Hybrid, and Multichip Module Package Design Guidelines</u>. New York: John Wiley & Sons, Inc., 1994.

88. Peeters, Joris, et al. "A Broad Band Loss Model for MCM Interconnections," <u>Proceedings 1993 IEEE Multi-Chip Module Conference</u>. Washington: IEEE Computer Society Press, 1993.

89. Rainal, Attilio J. "Reflections from Bends in a Printed Conductor." In Stuart K. Tewksbury, editor. <u>Microelectronics System Interconnections: Performance and Modeling</u>. New York: IEEE Press, 1994.

90. Romeo, Fabio and Mauro Santomauro. "Time-Domain Simulation of n Coupled Transmission Lines." In Stuart K. Tewksbury, editor. <u>Microelectronics System Interconnections: Performance and Modeling</u>. New York: IEEE Press, 1994.

91. Roy, Kaushik. "A Bounded Search Algorithm for Segmented Channel Routing for FPGA's and Associated Channel Architecture Issues," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. **12**:1695-1705. New York: IEEE Publishing, November 1993.

92. Rowlands, David O. Electrical Engineer, Special Purpose Processor Development Group, Mayo Foundation, Rochester Mn. Personal interview. 1-2 February 1995.

93. Sakurai, T. and K. Tamaru. "Simple Formulas for Two- and Three-Dimensional Capacitances." In Stuart K. Tewksbury, editor. Microelectronics System Interconnections: Performance and Modeling. New York: IEEE Press, 1994.

94. Sanaie, R., et al. "Integrating Subnetworks Characterized by Measured Data into Moment-matching Simulations," Proceedings 1994 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1994.

95. Sandborn, Peter A. and Hector Moreno. Conceptual Design of Multichip Modules and Systems. Boston: Kluwer Academic Publishers, 1994.

96. Sengupta, Maitreya, et al. "Crosstalk Driven Routing Advice," Electronics Components and Technology Conference, 1994.

97. Shouhei, Seki and Hideki Hasegawa. "Analysis of Crosstalk in Very High-Speed LSI/VLSI's Using a Coupled Multiconductor MIS Microstrip Line Model." In Stuart K. Tewksbury, editor. Microelectronics System Interconnections: Performance and Modeling. New York: IEEE Press, 1994.

98. Shrivastava, U.A. "Fast and Accurate Algorithms for Calculation of Self and Mutual Inductances of Rectangular Conductors," Proceedings 8th Annual International Electronics Packaging Conference. Wheaton: International Electronics Packaging Society Inc., 1988.

99. Simovich, Slobodan, et al. "Delay and Reflection Noise Macromodeling for Signal Integrity Management of PCBs and MCMs," Transactions CHMT, February, 1994.

100. Sriram, M. and S.M. Kang. "iPROMIS: An Interactive Performance Driven Multilayer MCM Router," Proceedings 1993 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1993.

101. Sugiuchi, Yo, et al. "Interconnect Optimization using Asymptotic Waveform Evaluation (AWE)," Proceedings 1994 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1994.

102. Tasker, Shiv. "Adopting MCM Design Technologies," <u>Computer Design</u>. **32**:82-86. Westford: Pennwell, June 1993.

103. Tripathi, Vijai K. and Richard J. Bucolo. "Analysis and Modeling of Multilevel Parallel and Crossing Interconnection Lines." In Stuart K. Tewksbury, editor. <u>Microelectronics System Interconnections: Performance and Modeling</u>. New York: IEEE Press, 1994.

104. Tsuk, Michael J. and Jin Au Kong. "A Hybrid Method for the Calculation of the Resistance and Inductance of Transmission Lines with Arbitrary Cross Sections." In Stuart K. Tewksbury, editor. <u>Microelectronics System Interconnections: Performance and Modeling</u>. New York: IEEE Press, 1994.

105. Van Petegem, Wim. "Electrothermal Simulation and Design of Integrated Circuits," <u>IEEE Journal of Solid-State Electronics</u>. **29**:143-146. New York: IEEE Press, February 1994.

106. Vemuri, Ranga, et al. "An Integrated Multicomponent Synthesis Environment for MCMs," <u>Computer</u>. 26:62-74. New York: IEEE Publishing, April 1993.

107. Vemuri, Ranga, et al. "DSS: A Distributed High-Level Synthesis System," <u>IEEE Design and Test of Computers</u>. New York: IEEE Publishing, June 1992.

108. Vemuri, Ranga, et al. "Experiences in Functional Validation of a High Level Synthesis System," <u>30th ACM/IEEE Design Automation Conference</u>. New York: IEEE Publishing, 1993.

109. Venkatachalam, P. N., et al. "Noise Containment in a High Wiring Density Multichip Module," <u>IEEE 2nd Topical Meeting on Electrical Performance of Electronic Packaging</u>. New York: IEEE Publishing, October 1993.

110. Walker, Charles S. <u>Capacitance, Inductance, and Crosstalk Analysis</u>. Boston: Artech House, 1990.

111. Wang, Taoyun, et al. "Quasi-Static Analysis of a Microstrip Via Through a Hole in a Ground Plane." In Stuart K. Tewksbury, editor. <u>Microelectronics System Interconnections: Performance and Modeling</u>. New York: IEEE Press, 1994.

112. Westbrook, Scott. "New Developments in Multichip Modules," <u>Computer Design</u>. **32**:77-78. Westford: Pennwell, June 1993.

113.	Wing, Omar and Hong Liu. "Lumped Approximation of the Characteristic Impedance of the RLGC Transmission Line with Error Analysis," Proceedings 1993 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1993.

114.	Xie, H. and Y.C. Lee. "Thermal Modeling Using Fuzzy Logic for 1-100 Watt/cm^2 Chips," ICEMM Proceedings, 1993.

115.	Xiong, G. J. "Algorithms for Global Routing," 23rd Design Automation Conference. pp. 824-830, 1986.

116.	Yu, Qiong, et al. "CD3D: A Constraint-Driven 3-Dimensional Router for Thick Film MCMs," International Conference and Exhibition: Multichip Modules. 2256:561-566. Denver: SPIE, 1994.

117.	Zhou, D., et al. "A Distributed-RLC Model for MCM Layout," Proceedings 1993 IEEE Multi-Chip Module Conference. Washington: IEEE Computer Society Press, 1993.

Vita

Captain Kenneth J. McClellan, Jr. was born 14 December 1966 in Mariemont, Ohio.

He graduated from Loveland Hurst High School in Loveland, Ohio in 1985 and attended

the U.S. Air Force Academy. He graduated with a Bachelor of Science in Electrical

Engineering (specialty: Digital Design) and a second major of applied physics in May

1989. Upon graduation, he received a regular commission in the USAF and served his

first tour of duty at Wright-Patterson AFB. Ohio. He began as a project officer for the

Productivity, Reliability, Availability, and Maintainability (PRAM) Program Office

where he managed several multi-million dollar projects until June 1990. He was then

assigned to work as a project engineer for a technology insertion program office. There

he was responsible for evaluating potential projects until entering the School of

Engineering, Air Force Institute of Technology, in June 1991. He graduated with a

Master of Science in Electrical Engineering (specialty: Electron Devices) in December

1992, and he stayed at AFIT to research Multichip Module routing. In March 1996, he

was assigned as a Test Technical Director in the Defense Nuclear Agency at Kirtland

AFB, New Mexico.

Permanent Address: 6504 Branch Hill-Miamiville Rd.

Loveland, Ohio 45140